Typy kolekcji

W tym momencie dużo się nauczyłeś! Możesz teraz utworzyć program, który przechowuje dane w stałych lub zmiennych i wykonuje na nich operacje, a także możesz sterować przepływem za pomocą warunków i pętli. Ale do tej pory głównie przechowywałeś pojedyncze wartości. Tu poznasz sposoby przechowywania zbiorów wartości. Swift ma trzy typy kolekcji: tablice, które przechowują uporządkowaną listę wartości; słowniki, które przechowują nieuporządkowaną listę par klucz-wartość; i zestawy, które przechowują nieuporządkowaną listę wartości. Pod koniec tego rozdziału dowiesz się, jak tworzyć tablice, słowniki i zestawy oraz jak wykonywać na nich operacje. Omówione zostaną następujące tematy:

• Zrozumienie tablic

• Rozumienie słowników

• Zrozumienie zestawów

Zrozumienie tablic

Załóżmy, że chcesz zapisać:

• Lista przedmiotów do kupienia w sklepie spożywczym

• Obowiązki, które musisz wykonywać co miesiąc

Do tego nadają się tablice. Tablica przechowuje wartości na uporządkowanej liście. Oto jak to wygląda:

Index	Value
0	value1
1	value2
2	value3

Wartości muszą być tego samego typu. Dostęp do dowolnej wartości w tablicy można uzyskać za pomocą indeksu tablicy, który zaczyna się od 0. Jeśli utworzysz tablicę za pomocą słowa kluczowego let, jej zawartość nie może zostać zmieniona po utworzeniu. Jeśli chcesz zmienić zawartość tablicy po utworzeniu, użyj słowa kluczowego var. Zobaczmy, jak pracować z tablicami. Utworzysz tablicę, przypisując jej wartość w następnej sekcji.

Tworzenie tablicy

W poprzednich rozdziałach utworzyłeś stałą lub zmienną, deklarując ją i przypisując jej wartość początkową. Tablicę można utworzyć w ten sam sposób. Wyobraź sobie, że twój współmałżonek poprosił cię o kupienie niektórych przedmiotów w sklepie spożywczym. Zaimplementujmy listę zakupów za pomocą tablicy. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

```
var shoppingList = ["Eggs", "Milk"]
```

Ta instrukcja tworzy zmienną tablicową o nazwie shoppingList. Przypisana wartość ["Jajka", "Mleko"] jest literałem tablicowym. Reprezentuje tablicę z dwoma elementami typu String, z „Jajkami” o indeksie 0. Użycie tutaj słowa kluczowego var oznacza, że zawartość tablicy może być modyfikowana. Ponieważ Swift używa wnioskowania o typie, elementy tej tablicy będą typu String. Wyobraź sobie, że

musisz sprawdzić, ile przedmiotów potrzebujesz w sklepie. W następnej sekcji dowiesz się, jak określić liczbę elementów w tablicy.

Sprawdzanie liczby elementów w tablicy

Aby dowiedzieć się, ile elementów jest w tablicy, użyj count. Wpisz i uruchom następujący kod:

shoppingList.count

Ponieważ tablica shoppingList zawiera dwa elementy, 2 jest wyświetlane w obszarze Wyniki. Możesz sprawdzić, czy tablica jest pusta, używając isEmpty. Wpisz i uruchom następujący kod:

shoppingList.isEmpty

Ponieważ tablica shoppingList zawiera dwa elementy, w obszarze Wyniki wyświetlane jest fałsz.

Wskazówka : Można również sprawdzić, czy tablica jest pusta, używając shoppingList. count == 0, ale użycie shoppingList.isEmpty zapewnia lepszą wydajność.

Wyobraź sobie, że twój współmałżonek dzwonił i pytał, czy możesz kupić kurczaka i olej do gotowania, kiedy jesteś w sklepie. W następnej sekcji zobaczysz, jak dodawać elementy na końcu tablicy i pod określonym indeksem tablicy.

Dodawanie nowego elementu do tablicy

Możesz dodać nowy element na końcu tablicy za pomocą append(_:). Wpisz i uruchom następujący kod:

shoppingList.append("Cooking Oil")

"Olej do gotowania" został dodany na końcu tablicy shoppingList, która zawiera teraz trzy elementy - „Jajka”, „Mleko” i „Olej do gotowania”. Można to zobaczyć w obszarze Wyniki.

Wskazówka: możesz również dodać nowy element do tablicy za pomocą operatora +, używając następującego kodu:

shoppingList = shoppingList + ["Cooking Oil"].

Możesz dodać nowy element w określonym indeksie za pomocą insert(_:at:). Wpisz i uruchom następujący kod:

shoppingList.insert("Chicken", at: 1)

Spowoduje to wstawienie „Kurczaka” w indeksie 1, więc teraz tablica shoppingList zawiera „Jajka”, „Kurczak”, „Mleko” i „Olej do gotowania”. Zauważ, że „Kurczak” jest drugim elementem tablicy, ponieważ pierwszy element ma indeks 0. Można to zobaczyć w obszarze Wyniki. Wyobraź sobie, że masz pierwszą pozycję na liście zakupów, a teraz musisz wiedzieć, jaka jest następna pozycja na liście. W następnej sekcji zobaczysz, jak uzyskać dostęp do określonego elementu tablicy za pomocą indeksu tablicy.

Dostęp do elementu tablicy

Możesz określić indeks tablicy, aby uzyskać dostęp do określonego elementu. Wpisz i uruchom następujący kod:

shoppingList[2]

Zwraca to element tablicy przechowywany pod indeksem 2, a w obszarze Wyniki wyświetlany jest napis „Mleko”. Wyobraź sobie, że twój współmałżonek zadzwonił i poprosił cię o mleko sojowe zamiast mleka. Ponieważ ta tablica została zadeklarowana przy użyciu słowa kluczowego var, możesz modyfikować przechowywane w niej wartości. Dowiesz się jak w następnej sekcji.

Przypisanie nowej wartości do określonego indeksu

Możesz zastąpić istniejący element tablicy, określając indeks i przypisując mu nową wartość. Wpisz i uruchom następujący kod:

```
shoppingList[2] = "Soy Milk"

shoppingList
```

Zastępuje to wartość zapisaną w indeksie 2 „Mleko” na „Mleko sojowe”. Tablica shoppingList zawiera teraz „Jaja”, „Kurczak”, „Mleko sojowe” i „Olej kuchenny”, jak pokazano w obszarze Wyniki. Pamiętaj, że użyty indeks musi być prawidłowy. Na przykład, nie możesz użyć indeksu 4, ponieważ jedynymi poprawnymi indeksami są tutaj 0, 1, 2 i 3. Spowodowałoby to awarię programu. Wyobraź sobie, że dzwonił twój współmałżonek i powiedział ci, że w lodówce jest kurczak, więc nie musisz już go kupować. W następnej sekcji zobaczysz dwa sposoby usuwania elementów z tablicy.

Usuwanie elementu z tablicy

Możesz usunąć element z tablicy, używając polecenia remove(at:). Wpisz i uruchom następujący kod:

```
shoppingList.remove(at: 1)

shoppingList
```

Spowoduje to usunięcie pozycji o indeksie 1, „Kurczak”, z tablicy shoppingList, więc teraz zawiera ona „Jaja”, „Mleko sojowe” i „Olej do gotowania”. Możesz to zobaczyć w obszarze Wyniki. Jeśli usuwasz ostatni element z tablicy, możesz zamiast tego użyć removeLast(). Wyobraź sobie, że masz wszystkie pozycje na liście i chciałbyś jeszcze raz przejrzeć listę, aby się upewnić. Będziesz musiał uzyskać dostęp do każdego elementu tablicy po kolei i wykonać operacje na każdym elemencie. Zobaczysz, jak to zrobić w następnej sekcji.

Iteracja po tablicy

Pamiętasz pętlę for-in, którą studiowałeś w poprzedniej części? Możesz go użyć do iteracji po każdym elemencie tablicy. Wpisz i uruchom następujący kod:

```
for shoppingListItem in shoppingList {

print(shoppingListItem)

}
```

To wypisuje każdy element tablicy w obszarze Debug. Możesz także użyć jednostronnych operatorów zasięgu. Są to operatory zakresów zawierające tylko wartość początkową, na przykład 1.... Wpisz i uruchom następujący kod:

```
for shoppingListItem in shoppingList[1...] {

print(shoppingListItem)

}
```

Spowoduje to wydrukowanie elementów tablicy, począwszy od elementu o indeksie 1 do obszaru Debug. Wiesz już, jak używać tablicy do tworzenia uporządkowanej listy, takiej jak lista zakupów, i jak wykonywać operacje na tablicach. W następnej sekcji przyjrzyjmy się, jak przechowywać nieuporządkowaną listę par klucz-wartość za pomocą słownika.

Zrozumienie słowników

Załóżmy, że piszesz aplikację Książka adresowa. Musisz przechowywać listę nazwisk i odpowiadających im numerów kontaktowych. Słownik byłby do tego idealny. Słownik przechowuje pary klucz-wartość na nieuporządkowanej liście. Oto jak to wygląda:

Key	Value
Key 1	Value 1
Key 2	Value 2
Key 3	Value 3

Wszystkie klucze muszą być tego samego typu i muszą być niepowtarzalne. Wszystkie wartości muszą być tego samego typu, ale niekoniecznie muszą być niepowtarzalne. Klucze i wartości nie muszą być tego samego typu. Używasz klucza, aby uzyskać odpowiednią wartość. Jeśli tworzysz słownik za pomocą słowa kluczowego let, jego zawartość nie może być zmieniona po utworzeniu. Jeśli chcesz zmienić zawartość po utworzeniu, użyj słowa kluczowego var. Zobaczmy, jak pracować ze słownikami. Słownik utworzysz, przypisując mu wartość w następnej sekcji.

Tworzenie słownika

Wyobraź sobie, że tworzysz aplikację Książka adresowa. W przypadku tej aplikacji będziesz używać słownika do przechowywania kontaktów. Podobnie jak tablicę, możesz utworzyć nowy słownik, deklarując go i przypisując mu wartość początkową. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

var contactList = ["Shah": "+60123456789", "Aamir": "+0223456789"]

Ta instrukcja tworzy zmienną słownikową o nazwie contactList. Przypisana wartość, ["Shah": "+60123456789", "Aamir": "+0223456789"], jest literałem słownikowym. Stanowi słownik składający się z dwóch elementów. Każdy element jest parą klucz-wartość, z nazwą kontaktu jako kluczem i numerem kontaktu jako wartością. Pamiętaj, że ponieważ nazwa kontaktu jest polem kluczowym, powinna być unikalna. Ponieważ słownik contactList jest zmienną, możesz zmienić zawartość słownika po jego utworzeniu. Zarówno klucz, jak i wartość są typu String ze względu na wnioskowanie o typie. Wyobraź sobie, że Twoja aplikacja musi wyświetlać całkowitą liczbę kontaktów. W następnej sekcji dowiesz się, jak określić liczbę elementów w słowniku.

Sprawdzanie liczby elementów w słowniku

Aby dowiedzieć się, ile elementów jest w słowniku, użyj count. Wpisz i uruchom następujący kod:

contactList.count

Ponieważ słownik contactList zawiera dwa elementy, 2 jest wyświetlane w obszarze Wyniki. Możesz sprawdzić, czy słownik jest pusty, używając isEmpty. Wpisz i uruchom następujący kod:

contactList.isPustyEmpty

Ponieważ słownik contactList składa się z dwóch elementów, w obszarze Wyniki wyświetlane jest fałsz.

Wskazówka : Można również sprawdzić, czy słownik jest pusty, korzystając z listy kontaktów. count == 0, ale użycie contactList.isEmpty zapewnia lepszą wydajność.

Wyobraź sobie, że właśnie zakończyłeś spotkanie i chcesz dodać nowy kontakt do swojej aplikacji. Ponieważ ten słownik został zadeklarowany przy użyciu słowa kluczowego var, możesz dodać do niego pary klucz-wartość. Dowiesz się jak w następnej sekcji.

Dodanie nowego elementu do słownika

Aby dodać nowy element do słownika, podaj klucz i przypisz mu wartość. Wpisz i uruchom następujący kod:

ContactList["Jane"] = "+0229876543"

contactList

Spowoduje to dodanie nowej pary klucz-wartość z kluczem „Jane” i wartością „+0229876543” do słownika contactList. Teraz składa się z „Shah”: „+60126789345”, „Aamir”: „+0223456789” i „Jane”: „+0229876543”. Możesz to zobaczyć w obszarze Wyniki. Wyobraź sobie, że chcesz zadzwonić do jednego ze swoich kontaktów i potrzebujesz numeru telefonu do tego kontaktu. W następnej sekcji zobaczysz, jak uzyskać dostęp do elementów słownika, określając klucz dla żądanej wartości.

Dostęp do elementu słownika

Możesz określić klucz słownika, aby uzyskać dostęp do odpowiadającej mu wartości. Wpisz i uruchom następujący kod:

contactList["Shah"]

Zwraca to wartość klucza „Shah”, a w obszarze Wyniki wyświetlany jest +60123456789. Wyobraź sobie, że jeden z Twoich kontaktów ma nowy telefon, więc musisz zaktualizować numer telefonu tego kontaktu. Możesz modyfikować pary klucz-wartość przechowywane w słowniku. Dowiesz się jak w następnej sekcji.

Przypisanie nowej wartości do istniejącego klucza

Możesz przypisać nową wartość do istniejącego klucza. Wpisz i uruchom następujący kod:

contactList["Shah"] = "+60126789345"

contactList

To przypisuje nową wartość do klucza „Shah”. Słownik contactList zawiera teraz „Shah”: „+60126789345”, „Aamir”: „+0223456789” i „Jane”: „+0229876543”. Możesz to zobaczyć w obszarze Wyniki. Wyobraź sobie, że musisz usunąć kontakt z aplikacji. Zobaczmy, jak możesz usunąć elementy ze słownika w następnej sekcji.

Usuwanie elementu ze słownika

Aby usunąć element ze słownika, przypisz nil do istniejącego klucza. Wpisz i uruchom następujący kod:

contactList["Jane"] = nil

contactList

Spowoduje to usunięcie elementu z kluczem „Jane” ze słownika contactList i zawiera teraz „Shah”: „+60126789345” i „Aamir”: „+0223456789”. Możesz to zobaczyć w obszarze Wyniki. Jeśli chcesz

zachować usuwaną wartość, użyj zamiast tego removeValue(for:Key). Wpisz i uruchom następujący kod:

var oldDictValue = contactList.removeValue(forKey: "Aamir")

oldDictValue

contactList

Spowoduje to usunięcie elementu z kluczem „Aamir” ze słownika contactList i przypisanie jego wartości do oldDictValue. Teraz oldDictValue zawiera teraz „+0223456789” i słownik contactList

zawiera „Shah”: „+60126789345”. Możesz to zobaczyć w obszarze Wyniki. Wyobraź sobie, że chciałbyś zadzwonić do każdego kontaktu, aby życzyć im szczęśliwego nowego roku. Będziesz musiał kolejno uzyskiwać dostęp do każdego elementu słownika i wykonywać operacje na każdym elemencie. Zobaczysz, jak to zrobić w następnej sekcji.

Iterowanie po słowniku

Podobnie jak w przypadku tablic, możesz użyć pętli for-in do iteracji po każdym elemencie słownika. Wpisz i uruchom następujący kod:

for (name, contactNumber) in contactList {

print("\(name) : \(contactNumber)")

}

To drukuje każdy element słownika w obszarze Debug. Ponieważ słowniki są nieuporządkowane, możesz otrzymać wyniki w innej kolejności po ponownym uruchomieniu tego kodu. Wiesz już, jak używać słownika do tworzenia nieuporządkowanej listy par klucz-wartość, takiej jak lista kontaktów, i jak wykonywać operacje słownikowe. W następnej sekcji zobaczmy, jak przechowywać nieuporządkowaną listę wartości w zbiorze.

Zrozumienie zbiorów

Załóżmy, że piszesz aplikację Filmy i chcesz przechowywać listę gatunków filmowych. Możesz to zrobić za pomocą zestawu. Zbiór przechowuje wartości na nieuporządkowanej liście. Oto jak to wygląda:

Value
Value 1
Value 2
Value 3

Wszystkie wartości są tego samego typu.

Jeśli utworzysz zbiór za pomocą słowa kluczowego let, jego zawartość nie może zostać zmieniona po utworzeniu. Jeśli chcesz zmienić zawartość po utworzeniu, użyj słowa kluczowego var. Zobaczmy, jak pracować z zestawami. Stworzysz zestaw, przypisując mu wartość w następnej sekcji.

Tworzenie zbioru

Wyobraź sobie, że tworzysz aplikację Filmy i chcesz przechowywać w niej gatunki filmowe. Jak widzieliśmy w przypadku tablic i słowników, zestaw można utworzyć, deklarując go i przypisując mu

nową wartość. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

var movieGenres: Set = ["Horror", "Action", "Romantic Comedy"]

Ta instrukcja tworzy zmienną zbioru o nazwie movieGenres. Zauważ, że przypisany do niego literał set, ["Horror", "Akcja", "Komedia romantyczna"] ma ten sam format, co literał tablicy, więc używasz adnotacji typu, aby ustawić typ movieGenres na Set. W przeciwnym razie wnioskowanie o typie Swifta utworzy zmienną tablicową, a nie zmienną zestawu. Użycie tutaj słowa kluczowego var oznacza, że zawartość zbioru może być modyfikowana. Elementy tego zestawu będą typu String ze względu na wnioskowanie o typie. Wyobraź sobie, że musisz pokazać całkowitą liczbę gatunków w swojej aplikacji. Zobaczmy, jak znaleźć liczbę elementów w zestawie w następnej sekcji.

Sprawdzanie ilości elementów w zbiorze

Aby dowiedzieć się, ile elementów jest w zbiorze, użyj licznika. Wpisz i uruchom następujący kod:

movieGenres.count

Ponieważ zestaw movieGenres zawiera trzy elementy, 3 jest wyświetlane w obszarze Wyniki. Możesz sprawdzić, czy zbiór jest pusty, używając isEmpty. Wpisz i uruchom następujący kod:

movieGenres.isEmpty

Ponieważ movieGenres zawiera trzy elementy, w obszarze Wyniki wyświetlane jest fałsz.

Wskazówka : Można również sprawdzić, czy zestaw jest pusty, używając movieGenres.count == 0, ale użycie movieGenres.isEmpty zapewnia lepszą wydajność.

Wyobraź sobie, że użytkownicy Twojej aplikacji mogą dodać do niej więcej gatunków. Ponieważ ten zbiór został zadeklarowany przy użyciu słowa kluczowego var, możesz dodawać do niego elementy. Dowiesz się jak w następnej sekcji.

Dodanie nowego elementu do zestawu

Możesz dodać nowy element do zbioru za pomocą insert(_:). Wpisz i uruchom następujący kod:

movieGenres.insert("War")

movieGenres

Dodaje to nowy element, „Wojna”, do zestawu filmów Gatunki, który zawiera teraz „Horror”, „Komedia romantyczna”, „Wojna” i „Akcja”. Jest to wyświetlane w obszarze Wyniki. Wyobraź sobie, że użytkownik chciałby wiedzieć, czy dany gatunek jest dostępny w Twojej aplikacji. Później dowiesz się, jak sprawdzić, czy element znajduje się w zbiorze.

Sprawdzanie, czy zbiór zawiera element

Aby sprawdzić, czy zestaw zawiera element, użyj zawiera(_:). Wpisz i uruchom następujący kod:

movieGenres.contains("War")

Ponieważ "Wojna" jest jednym z elementów zestawu movieGenres, true jest wyświetlane w obszarze Wyniki.

Wyobraź sobie, że użytkownik chce usunąć gatunek ze swojej listy gatunków. Zobaczmy, jak usunąć elementy z zestawu, które nie są już potrzebne w następnej sekcji.

Usuwanie przedmiotu z zbioru

Aby usunąć element z zestawu, użyj remove(_:). Usuwaną wartość można przypisać do zmiennej lub stałej. Jeśli wartość nie istnieje w zestawie, zostanie zwrócone nil. Wpisz i uruchom następujący kod:

var oldSetValue = movieGenres.remove("Action")

oldSetValue

movieGenres

„Action” została usunięta z zestawu movieGenres i przypisana do oldSetValue, a zestaw movieGenres zawiera teraz „Horror”, „Romantic Comedy” i „War”. Zobaczysz to wyświetlane w obszarze Wyniki. Aby usunąć wszystkie elementy z zestawu, użyj removeAll(). Wyobraź sobie, że chcesz wyświetlać wszystkie gatunki swojej aplikacji jako rekomendacje dla użytkowników aplikacji. Możesz iterować i wykonywać operacje na każdym elemencie zestawu. Zobaczmy, jak to zrobić w następnej sekcji.

Iteracja w zbiorze

Podobnie jak w przypadku tablic i słowników, możesz użyć pętli for-in do iteracji po każdym elemencie w zbiorze. Wpisz i uruchom następujący kod:

for genre in movieGenres {

print(genre)

}

Powinieneś zobaczyć każdy element zbioru w obszarze Debug. Ponieważ zbiory są nieposortowane, możesz otrzymać wyniki w innej kolejności po ponownym uruchomieniu tego kodu. Wyobraź sobie, że chcesz, aby Twoja aplikacja wykonywała operacje na gatunkach, które lubisz, z gatunkami, które lubi inna osoba. W następnej sekcji dowiesz się o różnych operacjach, które możesz wykonać ze zbiorami w Swift.

Eksplorowanie operacji na zbiorach

Łatwo jest wykonywać operacje na zbiorach, takie jak suma, przecięcie, odejmowanie i różnica symetryczna. Wpisz i uruchom następujący kod:

let movieGenres2: Set = ["Science Fiction", "War", "Fantasy"]

movieGenres.union(movieGenres2)

movieGenres.intersection(movieGenres2)

movieGenres.subtracting(movieGenres2)

movieGenres.symmetricDifference(movieGenres2)

Tutaj wykonujesz operacje na zbiorach na dwóch zestawach, movieGenres i movieGenres2. Zobaczmy wyniki każdej operacji na zbiorze:

• union(_:) zwraca nowy zbiór zawierający wszystkie wartości z obu zestawów, więc {"Horror", "Komedia romantyczna", "Wojna", "Science Fiction", "Fantasy"} będą wyświetlane w polu wyników .

• intersection(_:) zwraca nowy zbiór zawierający tylko wartości wspólne dla obu zbiorów, więc w polu Wyniki wyświetli się {"War"}.

• odejmowanie(_:) zwraca nowy zbiór bez wartości z podanego zestawu, więc w polu Wyniki wyświetli się {"Horror", "Komedia romantyczna"}.

• symmetricDifference(_:) zwraca nowy zbiór bez wartości wspólnych dla obu zestawów, więc w polu Wyniki zostanie wyświetlony {"Horror", "Komedia romantyczna", "Science Fiction", "Fantasy"}.

Wyobraź sobie, że chcesz, aby Twoja aplikacja porównywała gatunki, które lubisz, z gatunkami, które lubi inna osoba. W następnej sekcji dowiesz się, jak sprawdzić, czy zestaw jest równy innemu zestawowi, jest częścią innego zestawu lub nie ma nic wspólnego z innym zestawem.

Odkrywanie członkostwa w zbiorze i równości

Łatwo jest sprawdzić, czy zbiór jest równy, podzbiór, nadzbiór lub rozłączny inny zbiór. Wpisz i uruchom następujący kod:

```
let movieGenresSubset: Set = ["Horror", "Romantic Comedy"]

let movieGenresSuperset: Set = ["Horror", "Romantic Comedy",

"War", "Science Fiction", "Fantasy"]

let movieGenresDisjoint: Set = ["Bollywood"]

movieGenres == movieGenres2

movieGenresSubset.isSubset(of: movieGenres)

movieGenresSuperset.isSuperset(of: movieGenres)

movieGenresDisjoint.isDisjoint(with: movieGenres)
```

Zobaczmy, jak działa ten kod:

• Operator isEqual (==) sprawdza, czy wszystkie elementy jednego zbioru są takie same jak elementy innego zbioru. Ponieważ nie wszyscy członkowie zbioru movieGenres są tacy sami jak w zbiorze movieGenres, w obszarze Wyniki zostanie wyświetlona wartość false.

• isSubset(of:) sprawdza, czy zbiór jest podzbiorem innego zbioru. Ponieważ wszyscy członkowie zestawu movieGenresSubset znajdują się w zbiorze movieGenres, w obszarze Wyniki zostanie wyświetlona wartość true.

• isSuperset(of:) sprawdza, czy zbiór jest nadzbiorem innego zbioru. Ponieważ wszyscy członkowie zbioru movieGenres znajdują się w zbiorze movieGenresSuperset, w obszarze Wyniki zostanie wyświetlona wartość true.

• isDisjoint(with:) sprawdza, czy zbiór nie ma wartości wspólnych z innym zbiorem. Ponieważ zbiór movieGenresDisjoint nie ma elementów wspólnych z zbiorem movieGenres, w obszarze Wyniki zostanie wyświetlona wartość true.

Wiesz już, jak używać zbioru do tworzenia nieuporządkowanej listy wartości, takiej jak lista gatunków filmowych, i jak wykonywać operacje na zestawach. Na tym kończy się rozdział o typach kolekcji. Bardzo dobrze!

Podsumowanie

Przyjrzałeś się typom kolekcji w Swift. Najpierw dowiedziałeś się o tablicach. Pozwala to na użycie uporządkowanej listy wartości do reprezentowania pozycji, takiej jak lista zakupów, i wykonywania na

niej operacji. Następnie dowiedziałeś się o słownikach. Dzięki temu możesz używać nieuporządkowanej listy par klucz-wartość do reprezentowania elementu, takiego jak lista kontaktów, i wykonywania na nim operacji. Wreszcie dowiedziałeś się o zbiorach. Pozwala to na użycie nieuporządkowanej listy wartości do reprezentowania elementu, takiego jak lista gatunków filmowych, i wykonywania na nim operacji. Wkrótce nauczysz się grupować zestaw instrukcji za pomocą funkcji. Jest to przydatne, gdy chcesz wielokrotnie wykonać zestaw instrukcji w swoim programie.