Zintegrowany Program Rozwoju Politechniki Lubelskiej - część druga

POLITECHNIKA LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

KIERUNEK STUDIÓW INFORMATYKA

MATERIAŁY DO ZAJĘĆ LABORATORYJNYCH

Programowanie w języku SWIFT

Autor:
dr inż. Maria Skublewska-Paszkowska

Lublin 2021

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Unia Europejska Europejski Fundusz Społeczny

LABORATORIUM 4. INSTRUKCJE WARUNKOWE.

Cel laboratorium:

Poznanie instrukcji warunkowych w języku Swift.

Zakres tematyczny zajęć:

- instrukcja *if*,
- instrukcja *switch*,
- instrukcja *guard*.

Pytania kontrolne:

1. Jakie dostępne są instrukcje warunkowe w języku Swift?
2. Jak działa instrukcja *switch*?
3. W jaki sposób można dopasować daną do zakresu wartości w instrukcji *switch*?
4. Jak działają warunki *where* w instrukcji *switch*?
5. Omów działanie *break* oraz *fallthrough* w instrukcji *switch*.
6. Kiedy stosuje się instrukcję *guard*?

Instrukcje warunkowe (ang. *Conditional Statements*) pozwalają na sprawdzenie warunków, aby wykonać określony kod (np. obsługę błędów). W języku Swift można stosować instrukcję *if* lub *case*. Pierwszą z nich stosuje się do oceny prostych warunków z tylko kilkoma możliwymi wynikami. Druga instrukcja jest używana w bardziej złożonych sytuacjach, w szczególności przy dopasowaniu wzorców.

Instrukcja *if* wykonywana jest, jeśli podany warunek jest prawdziwy (ma wartość *true*). W przeciwnym razie wykonywane są instrukcje po słowie kluczowym *else*. Przykład sprawdzenia, czy liczba jest parzysta został przedstawiony na Listingu 4.1. Operator *%* oznacza operację modulo. Liczba jest losowana z przedziału 1-99. Niezależnie, ile instrukcji jest wykonywanych po danym warunku, należy stosować nawiasy klamrowe, grupujące te instrukcje.

Listing 4.1. Sprawdzenie czy liczba jest parzysta

```
let rand = Int.random(in: 1..<100)

if(rand % 2 == 0){
    print("Liczba \(rand) jest parzysta")
} else {
    print("Liczba \(rand) nie jest parzysta")
}
```

Zagnieżdżona instrukcja if została przedstawiona na Listingu 4.2. Sprawdzana jest, która cyfra została wylosowana po rzucie kostką sześcienną. Cyfra ta jest losowana z zakresu 1-6.

Listing 4.2. Sprawdzenie wylosowanej wartości

```
let a = Int.random(in: 1..<7)

if(a == 1){
    print("Wylosowano 1 ")
} else if (a == 2){
    print("Wylosowano 2 ")
} else if (a == 3){
    print("Wylosowano 3 ")
} else if (a == 4){
    print("Wylosowano 4 ")
} else if (a == 5){
    print("Wylosowano 5 ")
} else {
    print("Wylosowano 6 ")
}
```

Instrukcja *switch* stanowi alternatywę dla instrukcji *if*. Porównuje ona wartość do wzorca dopasowania. Jeśli nastąpi dopasowanie, wykonywany jest blok instrukcji zdefiniowany po wzorcu. Instrukcja *switch* porównuje tylko dane tego samego typu. Każda instrukcja *switch* posiada słowo kluczowe *case*, po którym następuje dopasowanie, a po dwukropku instrukcje, które mają zostać wykonane. Każda instrukcja *switch* musi zawierać wszystkie możliwie dopasowania. Przydatną opcją jest wartość domyślna (*default*), która jest wykonywana, gdy nie zostanie znalezione żadne dopasowanie. Przykład instrukcji dla pór roku został przedstawiony na Listingu 4.3.

Listing 4.3. Wybór pory roku

```
print("Podaj swoją ulubioną porę roku")
let str = readLine()
switch (str){
    case "wiosna":
                print("Twoją ulubioną porą roku jest wiosna.")
    case "lato":
                print("Twoją ulubioną porą roku jest lato.")
    case "jesień" :
                print("Twoją ulubioną porą roku jest jesień.")
    case "zima" :
                print("Twoją ulubioną porą roku jest zima.")
    default:
                print("Podałeś błędną porę roku")
}
```

Można zapewnić, aby dopasowanie brało pod uwagę różne wartości, należy je wymienić, oddzielając przecinkami (Listing 4.4).

Listing 4.4. Wybór pory roku o różnej pisowni

```
print("Podaj swoją ulubioną porę roku")
let str = readLine()
switch (str){
    case "wiosna", "Wiosna":
                print("Twoją ulubioną porą roku jest wiosna.")
    case "lato", "Lato":
                print("Twoją ulubioną porą roku jest lato.")
    case "jesień", "Jesień" :
                print("Twoją ulubioną porą roku jest jesień.")
    case "zima", "Zima" :
                print("Twoją ulubioną porą roku jest zima.")
    default: print("Podałeś błędną porę roku")
}
```

W instrukcji *switch* można sprawdzać, czy dana należy do danego przedziału. Należy zdefiniować zakresy po słowie kluczowym *case*. Na Listingu 4.5 przedstawiono instrukcję, która sprawdza, czy wczytana liczba całkowita jest jedno-, dwu-, trzy-, cztero-, czy pięciocyfrowa. Dla każdego przypadku zdefiniowano odpowiednie zakresy danych.

Listing 4.5. Dopasowanie liczby do zakresu

```
print("Podaj liczbę całkowitą z zakresu 1-99999:")
let num = Int(readLine()!)
switch (num!){
    case 1 ..< 10:
                print("Podałeś cyfrę.")
    case 10 ..< 100:
                print("Podałeś liczbę dwucyfrową.")
    case 100 ..< 1000:
                print("Podałeś liczbę trzycyfrową.")
    case 1000 ..< 10000:
                print("Podałeś liczbę czterocyfrową.")
    case 10000 ..< 100000:
                print("Podałeś liczbę pięciocyfrową.")
    default: print("Podałeś liczbę spoza zakresu")
}
```

Switch pozwala także na deklarowanie zmiennych oraz sprawdzenie warunków. Przykład dopasowujący nazwę państwa (Polski i jej sąsiadów) po numerze kierunkowym został przedstawiony na Listingu 4.6. W każdym dopasowaniu tworzona jest stała *x*, do której podstawiana jest wartość prefiksu dla poszczególnych państw przy użyciu słowa kluczowego *where*.

Listing 4.6. Sprawdzenie państwa po numerze kierunkowym

```
print("Podaj numer  telefonu dla Polski lub jej sąsiadów z
kierunkowym (w postaci np +48):")
let num = readLine()
switch (num!){
    case let x where x.hasPrefix("+48"):
                                print("Podałeś numer z Polski")
    case let x where x.hasPrefix("+49"):
                                print("Podałeś numer z Niemiec")
    case let x where x.hasPrefix("+420"):
                                print("Podałeś numer z Czech")
    case let x where x.hasPrefix("+421"):
                               print("Podałeś numer ze Słowacji")
    case let x where x.hasPrefix("+380"):
                               print("Podałeś numer z Ukrainy")
    case let x where x.hasPrefix("+375"):
                               print("Podałeś numer z Białorusi")
    case let x where x.hasPrefix("+370"):
                               print("Podałeś numer z Litwy")
    case let x where x.hasPrefix("+7"):
                               print("Podałeś numer z Rosji")
    default:
              print("Podałeś błędny numer")
}
```

Instrukcja *switch* nie pozwala na brak instrukcji po słowie kluczowym *case:*. Jeśli zachodzi taka potrzeba, można zastosować instrukcję *break*, która kończy całą instrukcję. Na Listingu 4.7 pierwsze dopasowanie nic nie wyświetla i kończy całą instrukcję. Można także użyć słowa kluczowego *fallthrough*, które oznacza, że po dopasowaniu, instrukcja *switch* nie jest kończona, ale wykonywane są instrukcje po kolejnym dopasowaniu. Na Listingu 4.8 instrukcję *break* zastąpiono *fallthrough*. Jeśli użytkownik poda 0, zostanie wyświetlony tekst po kolejnym dopasowaniu „*Jesteś niski*”.

Listing 4.7. Sprawdzenie wzrostu użytkownika

```
print("Podaj swój wzrost w cm")
let zad = Int(readLine()!)
switch (zad!){
    case 0..<100: break
    case 1..<100: print("Jesteś niski")
    case 100..<180: print("Jesteś średni")
    case 180..<210: print("Jesteś wysoki")
    default: print("Błędny wzrost")
}
```

Listing 4.8. Sprawdzenie wzrostu – instrukcja fallthrough

```
print("Podaj swój wzrost w cm ")
let zad = Int(readLine()!)
switch (zad!){
    case 0..<100: fallthrough
    case 1..<100: print("Jesteś niski")
    case 100..<180: print("Jesteś średni")
    case 180..<210: print("Jesteś wysoki")
    default: print("Błędny wzrost")
}
```

Instrukcja *guard*, podobnie jak instrukcja *if*, wykonuje instrukcje w zależności od wartości logicznej wyrażenia. Instrukcja ta jest stosowana do wykonania kodu, jeśli warunek jest nieprawdziwy. Dlatego zawsze zawiera klauzulę *else*. Wszelkie zmienne lub stałe, którym zostaną przypisane wartości przy użyciu opcjonalnego powiązania jako części warunku, są dostępne dla reszty bloku kodu. Przykład sprawdzenia, czy wczytana wartość z klawiatury jest liczbą całkowitą przedstawiono na Listingu 4.9. Jeśli łańcuch nie może zostać przekonwertowany na liczbę całkowitą, zostanie wyświetlony błąd informujący, że nie podano liczby całkowitej.

Listing 4.9. Sprawdzenie poprawności wczytanej wartości

```
print("Podaj liczbę całkowitą")
guard let num = Int(readLine()!) else {
    fatalError("To nie jest liczba całkowita")
 }
print(num)
```

Swift ma wbudowaną obsługę sprawdzania dostępności API, co gwarantuje, że interfejsy API, które są niedostępne w danym celu wdrożenia, nie zostaną użyte. Kompilator używa informacji o dostępności w zestawie SDK, aby sprawdzić, czy wszystkie interfejsy API używane w kodzie są dostępne w miejscu docelowym wdrożenia określonym przez projekt (Listing 4.10).

Listing 4.10. Sprawdzenie dostępności API

```
if #available(iOS 11, macOS 10.12, *) {
    print("Wersja dostępna")
} else {
    print("Wersja niedostępna")
}
```

Zadanie 4.1. Utwórz aplikację konsolową – rok przestępny

Polecenie 1. Napisz program, który wczyta od użytkownika rok i sprawdzi, czy jest on przestępny. Należy sprawdzić poprawność danych.

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.2. Utwórz aplikację konsolową – wiek

Polecenie 1. Napisz program, który dla wczytanej zmiennej całkowitej z zakresu <2,3010> oznaczającej rok, obliczy i wyświetli wiek, który reprezentuje wczytana dana.

Przykład:
rok 1000- wiek 10
rok 1903 – wiek 20

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.3. Utwórz aplikację konsolową – stypendium

Polecenie 1. Napisz program, który dla wczytanej średniej ze studiów, obliczy i wyświetli stypendium naukowe według wzoru:

$$S(\text{średnia}) = \begin{cases} \text{średnia} > 4{,}5 & \Rightarrow \quad 200 \\ \text{średnia} \in \langle 4{,}0;4{,}5) & \Rightarrow \quad 150 \\ \text{średnia} \in \langle 3{,}0;4{,}0) & \Rightarrow \quad 100 \\ \text{średnia} < 3{,}0 & \Rightarrow \quad 0 \end{cases}$$

Należy sprawdzić poprawność wprowadzonych danych.

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.4. Utwórz aplikację konsolową – kalkulator

Polecenie 1. Napisz program, który wczyta dwie liczby całkowite oraz wyświetla menu w postaci: ‘1-dodawanie, 2-odejmowanie, 3-mnożenie, 4-dzielenie’.
W zależności od wybranego elementu na ekranie powinien wyświetlić się wynik. Zabezpiecz odpowiednio program na wypadek dzielenia przez 0. Zastosuj instrukcję *switch*.

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.5. Utwórz aplikację konsolową – województwa Polski

Polecenie 1. Napisz program, który wczyta kod pocztowy, a następnie sprawdzi, czy jest to kod pocztowy województwa lubelskiego. Zastosuj instrukcję *switch*. Należy sprawdzić poprawność wprowadzonego kodu.

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.6. Utwórz aplikację konsolową – znak z klawiatury

Polecenie 1. Napisz program, który wczyta znak z klawiatury i określi, czy jest to samogłoska, spółgłoska, cyfra, czy inny znak. Zastosuj instrukcję *switch*.

Polecenie 2. Sprawdź działanie programu.

Zadanie 4.7. Utwórz aplikację konsolową – płeć

Polecenie 1. Napisz program, który wczyta pesel, określi i wyświetli, czy reprezentuje on kobietę, czy mężczyznę. Zastosuj instrukcję *switch*.

Polecenie 2. Sprawdź działanie programu.

Materiały zostały opracowane w ramach projektu
„Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga”,
umowa nr **POWR.03.05.00-00-Z060/18-00**
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020
współfinansowanego ze środków Europejskiego Funduszu Społecznego