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 2. UŻYCIE ZMIENNYCH I STAŁYCH.

Cel laboratorium:

Poznanie zmiennych i stałych języka Swift.

Zakres tematyczny zajęć:

- podstawy języka Swift,
- zmienne i stałe języka Swift,
- typy języka Swift,
- konwersja danych.

Pytania kontrolne:

1. Jakie cechy wyróżniają język Swift?
2. Czym różni się zmienna od stałej? Jak się deklaruje zmienne oraz stałe?
3. Omów polecenie wyświetlania danych.
4. Omów podstawowe typy danych. Jak można wyświetlić ich minimalną oraz maksymalną wartość?
5. Co oznacza Optionals i kiedy się je stosuje? W jaki sposób można wyświetlić wartość?

Swift to język programowania dla systemów takich jak iOS, macOS, watchOS i tvOS. Został wprowadzony na konferencji *Worldwide Developers Conference* w dniu 2 czerwca 2014. Jest następcą języka Objectove-C. Swift posiada więc cechy podobne do języków C oraz Objective-C. Zawiera takie typy: *Int, Double, Float, Bool, String*. Posiada jednak ich swoje wersje. Pozwala na definiowanie kolekcji typów takich jak: tablice (*Array*), zbiory (*Set*) oraz słowniki (*Dictionary*). Swift wprowadza także typy znane w Objective-C, jak krotki (*Tuples*).

Swift pozwala na definiowanie zmiennych, do których odwołuje się po nazwie oraz stałych, których wartość nie może zostać zmieniona. Stałe są stosowane do tworzenia bezpiecznego i czystego kodu.

Swift wprowadza opcjonalne typy (ang. *optional types*), które obsługują brak wartości. Oznacza to, że albo dana istnieje i ma znaną wartość, albo nie ma żadnej przypisanej wartości. Działają one na wszystkich typach i są jednym z kluczowych elementów języka.

Swift jest językiem bezpiecznego typu (ang. *type-safe language*), co oznacza, że pomaga on upewnić się, co do typu wartości. Jeśli zmienna wymaga podania wartości tekstowej, język zapobiegnie przypisaniu jej innego typu wartości. Mechanizm ten zapobiega także przekazaniu opcjonalnego typu do kodu, który wymaga nieopcjonalny typ. Pomaga także lokalizować błędy i je naprawiać.

Zmienne i stałe

Zmienne i stałe definiowane są za pomocą nazw z przypisanym konkretnym typem. Wartość stałej ustawiana jest raz i nie może zostać zmieniona w trakcie działania programu. Natomiast wartość zmiennej może zostać modyfikowana.

Zmienne i stałe muszą zostać zadeklarowane przed ich użyciem. Zmienne deklarowane są z użyciem słowa kluczowego *var*, a stałe z zastosowaniem *let* (Listing 2.1). Można także deklarować wiele zmiennych w jednej linii, oddzielając każdym przecinkiem.

Listing 2.1. Deklaracja stałej i zmiennej

```
let pi = 3.1415
var maxValue = 20
var a = 0.0, b = 1.0, c = 5.0
```

Typy podczas deklaracji

Podczas deklaracji zmiennych i stałych można także podać ich typ, aby wyraźnie sprecyzować rodzaj przechowanych danych. Typ podaje się po nazwie zmiennej lub stałej, oddzieloną dwukropkiem i po spacji (Listing 2.2). Takiej zmiennej w programie można przypisać odpowiednią wartość. Wiele zmiennych tego samego typu może zostać zadeklarowanych, których nazwy są oddzielone przecinkami.

Listing 2.2. Deklaracja zmiennej z typem

```
var name: String
name = "Ann"

var a, b, c: Double
```

W praktyce rzadko stosuje się typy. Jeśli podawana jest wartość inicjująca dla zmiennej lub stałej, Swift sam zainicjuje typ.

Nazwy zmiennych mogą zawierać prawie wszystkie znaki, włączając znaki Unicode. Nie mogą zawierać białych spacji, symboli matematycznych, strzałek, czy symboli rysunkowych. Nie mogą rozpoczynać się od cyfry, ale poza pierwszym znakiem, mogą zawierać cyfrę w dowolnym miejscu. Nazwy zmiennych muszą być unikalne w obrębie programu. Należy także unikać stosowania słów kluczowych języka Swift. Przykłady nazw zmiennych zostały przedstawione na Listingu 2.3.

Listing 2.3. Przykłady nazw zmiennych

```
var value1: Int
var 😀 = "smile"
```

Wyświetlanie zmiennych i stałych odbywa się przy pomocy funkcji *print(_:separator:terminator:)*. Można wyświetlić jedną lub wiele zmiennych/ stałych (Listing 2.4). W Xcode funkcja ta wyświetla dane konsolowo. Domyślnie funkcja kończy wyświetlanie przejściem do nowej linii. Aby pozostać w tej samej linii, należy zastosować opcję *terminator*. Na Listingu 2.5 dwa ostatnie linie kodu zostaną wyświetlone w jednej linii.

Listing 2.4. Wyświetlenie zmiennych i stałych

```
print("Hello word!")
var name: String
var age: 23
name = "Ann"
print(name)
print(name, age)
```

Listing 2.5. Wyświetlenie danych z i bez przejścia do nowej linii

```
print("Tekst z przejściem do nowej linii")
print("Tekst bez łamania linii", terminator:"")
print(" kolejna linia")
```

Aby wyświetlić tekst razem ze zmiennymi należy zastosować zmienną lub stałą jako symbol zastępczy, co zostało przedstawione na Listingu 2.6.

Listing 2.6. Wyświetlenie tekstu razem z wartościami zmiennych

```
var name = "Ania"
var age  = 23
print("Witaj \(name)! Masz \(age) lat")
```

W języku Swift komentarze mogą być jednowierszowe (//) lub wielowierszowe (/*…*/). W przeciwieństwie do języka C, można stosować zagnieżdżone wielowierszowe komentarze (Listing 2.7). Pozwalają one komentować duże bloki kodu.

Listing 2.7. Komentowanie

```
//jednowierszowy komentarz

/* Zewnetrzny komentarz
     /* Wewnetrzny
          komentarz
     */
*/
```

Język Swift nie wymaga stosowania średników kończących instrukcje. Stosowanie ich nie jest jednak błędem.

Typy całkowite

Typy całkowite stosowane są dla liczb ze znakiem (ang. *signed*): dodatnich, równych zero lub ujemnych oraz dla liczb bez znaku (ang. *unsigned*): dodatnich lub równych zero. Swift zapewnia te liczby dla formatu 8, 16, 32 i 64-bitowych. Posiadają one konwersje podobną do języka C: *UInt8* – 8-bitowej dla bez znaku, czy *Int32* – 32-bitowej ze znakiem. Minimalną oraz maksymalną wartość dla każdego typu można ustawić z zastosowaniem właściwości *min* oraz *max*, co przedstawiono na Listingu 2.8.

Listing 2.8. Minimalna i maksymalna wartość typu całkowitego

```
let minValue = Int16.min
let maxValue = Int16.max
```

Często stosowany jest typ całkowity *Int*, który przyjmuje rozmiar zgodny z rozmiarem platformy: *Int32* dla 32-bitowej lub *Int64* dla 64-bitowej. Jeśli nie trzeba stosować specyficznego typu, zaleca się stosowanie *Int*.

Typy zmiennoprzecinkowe

Swift zapewnia dwa typy zmiennoprzecinkowe: *Float* dla reprezentowania 32-bitowych liczb oraz *Double* dla 64-bitowych. Typ *Double* ma precyzję 15 liczb po przecinku, podczas gdy *Float* – 6. W przypadku, gdy zmienna lub stała zostanie zadeklarowana przy użyciu danej wartości, zostanie przypisany domyślny typ *Double*.

Aby wyświetlić liczbę zmiennoprzecinkową w odpowiednim formacie, należy utworzyć tekst w odpowiednim formacie, co przedstawiono na Listingu 2.9.

Listing 2.9. Formatowanie wyświetlania liczby zmiennoprzecinkowej

```
var a, b, c: Double
a = 878.66
b = 66.983
c = a/b
let str = String(format: "%.2lf", c)
print(str) //13.12
```

Literały liczbowe dla liczb całkowitych:

- liczba dziesiętna bez prefiksu,
- liczba binarna z prefiksem 0b,
- liczba ósemkowa z prefiksem 0o,
- liczba szesnastkowa z prefiksem 0x.

Na Listingu 2.10 przedstawiono liczby całkowite równe 21 w 4 literałach.

Listing 2.10. Literały znakowe dla liczby 21

```
let val = 21
let valBin = 0b10101
let valoct = 0o25
let valHex = 0x15
```

Literały dla liczb zmiennoprzecinkowych mogą być dziesiętne bez prefiksu oraz szesnastkowe z prefiksem 0x. Część dziesiętna opcjonalnie może mieć wykładnik poprzedzony *e*, a dla liczb szesnastkowych *p* (Listing 2.11).

Listing 2.11. Literały dla liczb zmiennoprzecinkowych

```
let val1 = 1.45e2 //145
let val2 = 1.45e-2 //0.0145
let valhex = 0xFp3 //120
```

Literały numeryczne mogą posiadać formatowanie, aby liczby były bardziej czytelne. Mogą być poprzedzone zerami, czy zawierać podkreślenia (Listing 2.12).

Listing 2.12. Literały z formatowaniem

```
let num1 = 00056.78
let num2 = 23_560_000
```

Konwersja liczb całkowitych i zmiennoprzecinkowych polega na rzutowaniu z jednego typu na drugi. Tym, na który wartość jest zmieniana, musi być jednoznacznie określony. Konwersja z liczb zmiennoprzecinkowych na całkowite została przedstawiona na Listingu 2.13, a z liczb całkowitych na zmiennoprzecinkowe na Listingu 2.14. Dzieląc 2 liczby całkowite, wynik także będzie liczbą całkowitą. Aby uzyskać wynik zmiennoprzecinkowy, obie liczby należy poddać konwersji na typ zmiennoprzecinkowy.

Listing 2.13. Konwersja na liczby całkowite

```
let num1 = 7.896543
let intNum1 = Int(num1)//7
let num2 = -5.63214
let intNum2 = Int(num2) //-5
```

Listing 2.14. Konwersja na liczby zmiennoprzecinkowe

```
let a = 5
let b = 2
let c1 = a / b //2
let c2 = Double(a) / Double(b) //2.5
```

Typ logiczny

Swift posiada podstawowy typ logiczny, *Bool*, który przyjmuje 2 wartości: *true* oraz *false*. Stosuje się je w instrukcjach warunkowych i sterujących.

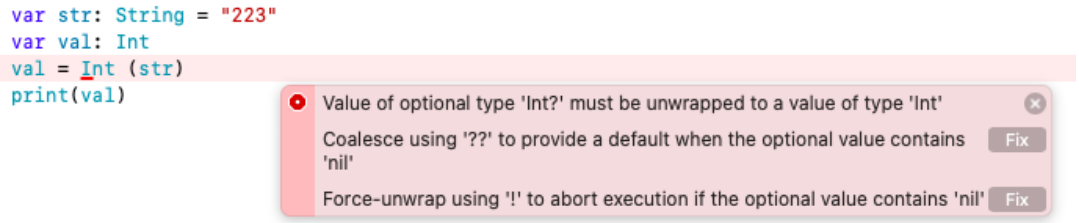
Optionals

Opcjonalność (ang. *Optionals*) stosowana jest w sytuacji, gdy wartość może nie być podana. Reprezentuje to dwa podejścia:

- jeśli wartość istnieje, opcjonalność zostaje odpakowana (ang. *unwrap*), aby uzyskać dostęp do niej;
- wartość nie została zdefiniowana.

Opcjonalność stosuje się m.in. w następujących przypadkach:

- konwersja z ciągu na liczbę nie zawsze może być wykonana (Rys. 2.1) – jeśli tekst nie może zostać przekonwertowany na liczbę, operacja zwraca wartość *nil*;
- właściwości kontroler widoku mają wartość *nil* przed załadowaniem;
- właściwości struktury, które mają wartość *nil*, dopóki jej wartości nie zostaną pobrane.



Rys. 2.1. Błędna konwersja tekstu na liczbę całkowitą

Optionals polega na dodaniu ? do typu, co przedstawiono na Listingu 2.15. Wynikiem działania takiego programu będzie: *Optional("223")*.

Listing 2.15. Przykład zastosowania Optionals

```
var str: String? = "223"
print(str)
```

Aby uzyskać dostęp do samej wartości, należy odpakować opcjonalność (ang. *unwrap an optional*). Można to zrobić na 4 sposoby:

- wymusić stosując !;
- za pomocą *optional binding* stosując instrukcję *if let*;
- za pomocą niejawnego *optional binding* stosując !;
- za pomocą optional chaining stosując *a?.b?.c*.

Dodanie znaku ! przy wyświetleniu zmiennej spowoduje wyświetlenie jej wartości (Listing 2.16).

Listing 2.16. Przykład zastosowania Optionals oraz wymuszonego rozpakowania

```
var str: String? = "223"
print(str!)
```

Na Listingu 2.17 przedstawiono poprawną konwersję tekstu na liczbę całkowitą. Ze względu, że konwersja może nie być możliwa do wykonania należy zastosować *Optionals* dla zmiennej całkowitej. Przy wyświetlaniu można użyć wymuszone rozpakowanie.

Listing 2.17. Konwersja tekstu na liczbę całkowitą

```
var str: String = "223"
var val: Int?
val = Int (str)
print(val!)
```

Zadanie 2.1. Utwórz aplikację konsolową – obliczenie wieku

Polecenie 1. Napisz program konsolowy, który wczyta rok urodzenia osoby, obliczy i wyświetli, ile osoba ma lat („*Masz …. lat*”). Bieżący rok należy ustawić jako stałą.
Do wczytywania służy funkcja: *readLine()*. Należy zastosować konwersję z typu tekstowego na typ całkowity. Należy zastosować Optionals.

Polecenie 2. Sprawdź działanie programu.

Zadanie 2.2. Utwórz aplikację konsolową – obwód i pole koła

Polecenie 1. Napisz program konsolowy, który wczyta promień koła, obliczy i wyświetli jego obwód oraz jego pole. Liczbę pi należy ustawić jako stałą. Promień, pole i obwód koła należy wyświetlić z zastosowaniem jednego polecenia. Należy liczby zmiennoprzecinkowe sformatować do dwóch miejsc po przecinku.

Polecenie 2. Sprawdź działanie programu.

Zadanie 2.3. Utwórz aplikację konsolową – pole całkowite i objętość sześcianu

Polecenie 1. Napisz program konsolowy, który wczyta bok sześcianu, obliczy i wyświetli jego pole całkowite oraz objętość. Wynik należy wyświetlić z zastosowaniem jednego polecenia oraz liczby zmiennoprzecinkowe sformatować do dwóch miejsc po przecinku.

Polecenie 2. Sprawdź działanie programu.

Zadanie 2.4. Utwórz aplikację konsolową – listwa

Polecenie 1. Pokój w kształcie prostokąta ma wymiary *a x b* metrów, a korytarz *z x b*. Pokój ma jedne drzwi prowadzące na korytarz, który na końcu posiada drzwi do kolejnego pomieszczenia. Napisz program, który obliczy, ile potrzeba metrów bieżących listwy do położenia w pokoju i korytarzu oraz cenę listwy. Należy pominąć drzwi o szerokości s metrów. Cenę listwy za metr bieżący oraz wszystkie wymiary podaje użytkownik.

Polecenie 2. Sprawdź działanie programu.

Zadanie 2.5. Utwórz aplikację konsolową – średnia

Polecenie 1. Napisz program, który obliczy średnią arytmetyczną 3 wygenerowanych losowo liczb całkowitych z zakresu od 1 do 9. Wyświetl te liczby oraz obliczoną średnią.

Polecenie 2. Sprawdź działanie programu.

Zadanie 2.6. Utwórz aplikację konsolową – średnia ważona

Polecenie 1. Napisz program, który wczyta 3 oceny uczniów z wagami (od 1 do 3 jako liczba całkowita). Oblicz ważoną średnią tych ocen.

Polecenie 2. Sprawdź działanie programu.

Zintegrowany Program Rozwoju Politechniki Lubelskiej - część druga

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