National Yang Ming Chiao Tung University, Department of Electrical Engineering

嵌入式系統設計
Embedded System Design

Lab 4: Dictionary, Reading and Writing Data

蕭安紘助教製作

一. 實驗目的

了解 Dictionary 的使用方法，並使用 table view 呈現 Dictionary 的資料。並且練習如何將 Dictionary 轉成想要的格式寫入檔案，使記帳程式紀錄的資料可以持續存在。

二. 實驗需求

環境：macOS 13.4 或以上
IDE：Xcode 14.0 或以上
語言：Swift 5

三. Dictionary

Dictionary 是 swift 中儲存 key-value pair 的資料型態，它儲存了 key 與 value 物件的對應關係。每一個 Dictionary 裡的 value 都會對應到 unique (唯一) 的 key 物件，根據 key ，dictionary 可以找到 key 對應到的 value 的值。在 dictionary 中，key 是唯一的，value 則可以重複，可以不同的 key 對應到同一個 value。

在使用上，如果有需要根據不同的 identifier (標籤)找到對應的物件或數值，可以考慮使用 Dictionary。如同現實生活中的字典一樣，根據我們想要查尋的單字(key)，即可找到對應的單字定義(value)。不過與現實生活中的字典不一樣的是 dictionary 中 key 之間沒有先後順序的關係，如果用 for-in iterate 不一定會拿到預期的順序。

1. Dictionary 的表示方法

Dictionary<Key, Value> 或 [Key: Value] 表示

```
var dict1: Dictionary<String,Double>
var dict2: [String:Double]
```

- dict1 和 dict2 的 type 都一樣，[String:Double] 這種方式為常用的表示法
- String, Int 都是常見可當作 key 的類別
- Value type 是 Any，即可存所有類別的value，使不同 key 可以對應到不同類型的資料。例如 "Date" 對應到 "3/20", "Temperature" 對應到 26.7。

2. 創建 Dictionary

(1) 空 dictionary

```
var dict1 = Dictionary<String,Double>()
var dict2 = [String:Double]()
```

(2) 直接給定dictionary的值

```
var dict = ["A":10,"B":20,"C":30,"D":40]
```

(3) 如果需要直接創建不同類型的 key 或 value 的 dictionary，須在最後補上 key 或 value 的類型描述

```
var dict = ["Date":"3/20","Temp":20.5] as [String : Any]
```

3. Dictionary 的存取與修改

(1) 取值

```
var dict = ["A":10,"B":20,"C":30,"D":40]
var value1 = dict["A"] // 10
var value2 = dict["E"] // nil
```

- 須注意如果 access 沒有存在的 key ，會回傳 nil
- 所以上方 value1, value2的 type 皆為 optional 的 Int?

(2) 長度

```
var dict = ["A":10,"B":20,"C":30,"D":40]
dict.count   // "4"
```

(3) 檢查是否為空

```
dict.count == 0
dict.isEmpty
```

- 兩個 statement 等價

(4) 取得所有 dictionary 擁有的 key ，並且轉成 array

```
var keyArray = Array(dict.keys)
```

(5) 修改與新增 item

```
var dict = ["A":10,"B":20,"C":30]
dict["B"] = 22  //["A":10,"B":22,"C":30]
dict["D"] = 50  //["A":10,"B":22,"C":30,"D",50]
```

(6) 移除 item

```
dict.removeValue(forKey: "A")
dict["B"] = nil
```

- dictionary 中 value 不能為 nil，如果設定某個 item 的 value 為 nil ，相當於移除這個 item

四. Type Casting

前述提到 Dictionary 可以使用 Any 使得 value 可以對應到多種 type 的物件。但如此一來，要使用 key 取值時，反而會無法確定取得的值是什麼型態。 此時，就需要使用 Type Casting 來確認物件是什麼型態

1. is Operator

   確認物件是否為某個形態

   ```
   var value:Any = "123"
   value is String //false
   ```

   ※ Lab 3 所說，protocol 也可以當成是一種 type ，因此 is 也可以用來判斷是否符合某個 protocol 。

2. as Operator

   將物件轉成特定的 Type，或是可以解釋成：「將這個物件當這個 type 來使用」，因為可能會失敗，所以又可以分成 as? (條件轉型) 與 as! (強制轉型) 兩種用法。條件轉型 as? 如果轉型失敗，會回傳nil。

   ```
   var value:Any = "123"
   value as? String    // String "123"
   value as! String    // String "123"
   value as? Int       // String nil
   value as! Int       // error
   ```

五. 讀寫檔案

在 iOS 系統內，要存下 App 執行時所產生的資料在硬碟內，達到 Data Persistent 有很多方法，例如使用 NSCoding 或是直接使用資料庫 Core Data，這些方法雖然好用，也較易重新讀取，不過卻無法直接與其他系統相通。因此在本次實驗，我們會使用最基本的檔案讀寫來純取資料

1. 取得 App 專用資料夾

   因為 iOS 的 sand box 安全機制，每個 app 都只能存取專屬的獨立儲存空間，我們可以使用 FileManager 來快速的取得 app 專用資料夾中，專門儲存 user 資料用的資料夾路徑。

```
let dir = FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask).first
```

2. 檔案路徑

   取得型態為 URL 的 app 專屬 user 資料夾路徑後，我們可接著生成要讀取的檔案位置之 URL

```
let fileURL = dir.appendingPathComponent(fileName)
```

3. 寫入檔案

   我們可以使用 Swift 的 String 本身提供的 write 函數來將檔案寫入到硬碟上，會取代掉原本的檔案內容。由於寫入可能會失敗，因此，我們需要用上 try and catch，來 handle 寫入錯誤時的狀況，避免因為寫入錯誤而造成程式 crash。

```
 do{
     try writeString.write(to: fileURL, atomically: true,
encoding: .utf8)
 }catch
 {
     print("write error")
 }
```

4. 讀取檔案

   有了檔案的 URL 後，我們也可以透過 String 提供的建構子來從檔案讀取出內容存 string 中，一樣，需要使用 try and catch。

```
do{
    try readString = String.init(contentsOf: fileURL, encoding: .utf8)

}catch
{
    print("read error")
}
```

※寫入與讀取的編碼必須要一樣。

六. 補充指令

1. Date 與 String 的轉換

   我們使用 DateFormatter 做 Date 與 String 間的轉換

```
//取得現在時間
let currentDate = Date()

//宣告要拿來轉換時間的formatter
let formatter = DateFormatter()

//設定字串轉換要用的格式
formatter.dateFormat = "yyyy/MM/dd hh:mm:ss"

//轉換時間成字串
let dateString = formatter.string(from: currentDate)
// "2018/03/22 09:32:13"


let otherDateString = "2018/03/11 12:07:53"

//轉換字串回時間
let date = formatter.date(from: otherDateString)
//"Mar 11, 2018 at 12:07 AM"
```

1. 字串的切分與組合

   - 把 string 用某個 separator 分割成 array

```
let string = "12_34_56"
let array = string.components(separatedBy: "_")  // ["12", "34", "56"]
```

   - 把 array 用某個 separator 組合成 string

```
let array = ["A","B","C","D"]
let string = array.joined(separator: "-") // "A-B-C-D"
```

※詳細的字串表示格式可以看參考 www.unicode.org/reports/tr35/tr35-31/tr35-dates.html#Date_Format_Patterns

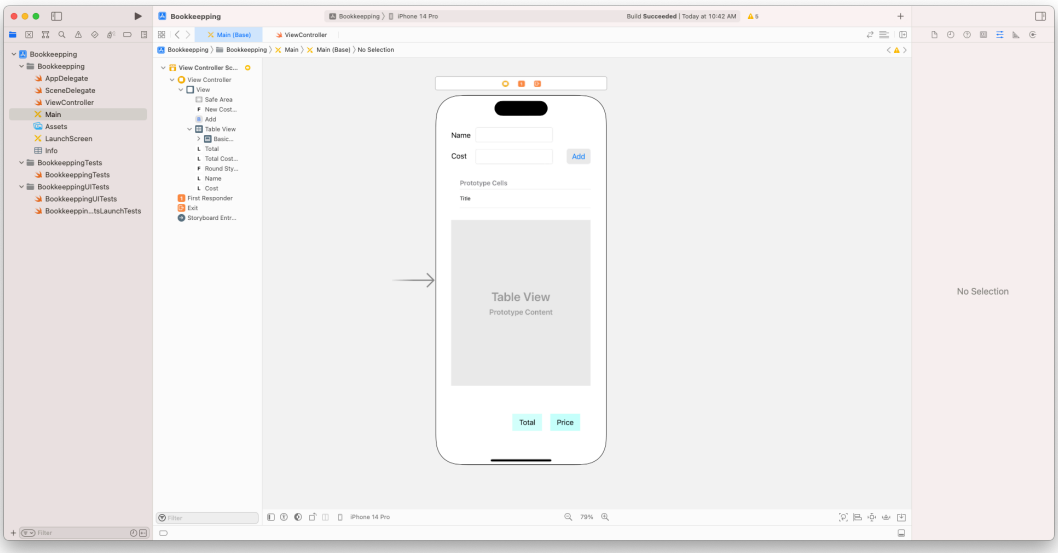
七. 實驗步驟

1. 準備基礎 Bookkeeping 專案

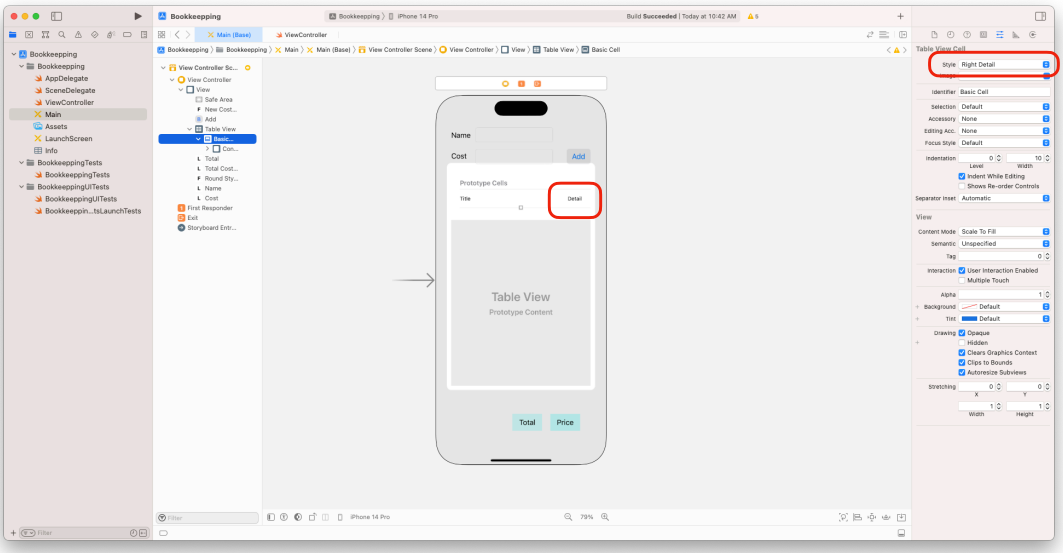
(1) 請依照 Lab 3 準備好基本的 Bookkeeping 專案

2. 增加 Bookkeeping 資料輸入介面

(1) 將介面增加新的 Name 輸入 text field



(2) 將 prototype cell 改成可以顯示 name 與 cost **兩個欄位**的樣式，選取 table view 上的 cell ，在右方 attributes inspector 將 style 改成 right detail 或其他格式



(3) 在 view controller 新增 name text field 的 outlet

```
    @IBOutlet weak var nameField: UITextField!
```

3. 將儲存資料的型態改成 Dictionary

(1) 將 dataArray 變數改成 array of dictionary [String:Any]

```
    //空的 [String:Any] dictionary array 來儲存消費資料
    var dataArray = [[String:Any]]()
```

(2) 更改新增資料的格式，每次新增包含 date, name, cost 三種欄位的資料到 array

```
    @IBAction func addData(_ sender: Any) {

                                    ...

        //檢查輸入的文字可不可以轉成 Double，如果不能，離開function
        guard let newCost = Double(newCostString) else { return }

        //檢查、取得輸入的名字
        guard let newName = nameField.text, !newName.isEmpty else
{ return }

        //取得輸入資料當下的時間
        let newDate = Date()

        //創造新的Dictionary加入array

dataArray.append(["name":newName,"cost":newCost,"date":newDate])

                                    ...

        //準備下次輸入，將輸入匡清空
        newCostField.text = ""
        nameField.text = ""
                                    ...

    }
```

※在此我們儲存 date 只是為了記錄用，沒有要顯示在介面上

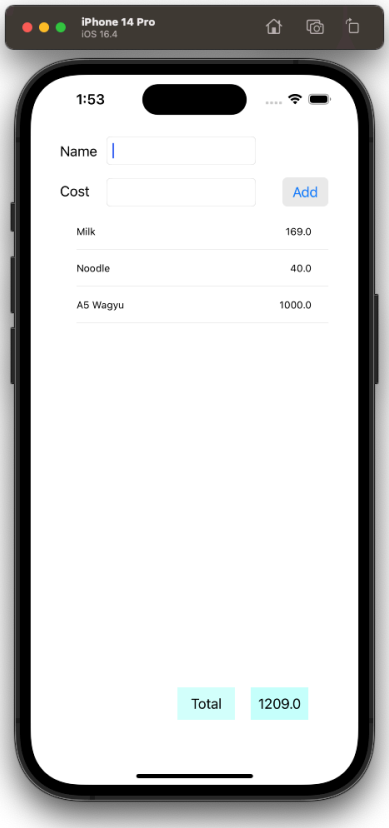
(3) 更改每個 row 該如何顯示資料

```
    func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {
                                    ...
        //取得dictionary取得要顯示的資料的key的值

        //取得key為name的資料條件轉型成String
        //如果沒有這個key value pair 或轉型不成功，使用"No name"字串取代
        let name = dataArray[indexPath.row]["name"] as? String ?? "No
name"

        //取得key為cost的資料條件轉型成Double
        //如果沒有這個key value pair 或轉型不成功，使用0.0取代
        let cost = dataArray[indexPath.row]["cost"] as? Double ?? 0.0

        //設定cell的內容
        //把name設定到cell的title
        cell.textLabel?.text = name

        //把cost設定到cell的detail title
        cell.detailTextLabel?.text = "\(cost)"

        return cell
    }
```

(4) 修改計算總額的程式碼，計算 dictionary array 的 cost 的總額

※請自行完成

(5) 執行 app 測試修改是否成功



4. 儲存，讀取資料

(1) 為了有較乾淨的程式碼，我們將讀檔與寫檔的功能寫成各自的 method，將字串寫入檔案的 method 需要有 file name 及要寫入的 string 兩種 input

```
    func writeStringToFile(writeString:String, fileName:String) {
        //取得app專用資料夾路徑，並且確定檔案路徑存在
        guard let dir =
FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask).first else{ return}
        //在路徑後加上檔名，組合成要寫入的檔案路徑
        let fileURL = dir.appendingPathComponent(fileName)
        do{
            //嘗試使用utf8格式寫入檔案
            try writeString.write(to: fileURL, atomically: false,
encoding: .utf8)

        }catch
        {
            //若寫入錯誤print錯誤
            print("write error")
        }
    }
```

※我們可使用 print("\(fileURL)") 來查看檔案實際上在電腦上的路徑

(2) 將檔案讀出成字串的 method 只需要 input 檔名，return 讀取出來的 string

```
    func readFileToString(fileName:String) -> String {
        //取得app專用資料夾路徑，並且確定檔案路徑存在，如果不存在，return空字串
        guard let dir = FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask).first else{
            return ""
        }
        //在路徑後加上檔名，組合成要讀取的檔案路徑
        let fileURL = dir.appendingPathComponent(fileName)
        //宣告要儲存讀取出來的string的變數
        var readString = ""
        do{
            //嘗試使用utf8格式讀取字串
            try readString = String.init(contentsOf: fileURL, encoding: .utf8)
        }catch

        {
            //若讀取錯誤print錯誤
            print("read error")
        }

        //return讀取出的string
        return readString
    }
```

(3) **完成儲存 Dictionary Array 的 function**。我們需要把字典檔轉成字串的格式，在此，我們選擇使用 CSV (Comma-separated values) 格式儲存。CSV 格式為每個 row 為一筆資料，每個 column 為一個欄位，欄位間用 "," 隔開。範例如下：

```
2018/12/22,"food1",120
2018/12/22,"drink1",30
2018/12/24,"food2",70
2018/12/25,"food3",90
2018/12/26,"drink2",45
```

我們可以把將 Dictionary 轉成csv字串格式，並且寫入檔案的功能包成一個 method，請自行將此 method 補完。可以參考前面 Date 與 String 的轉換。

```
    func saveDataArray(){

        //宣告儲存最後string的變數
        var finalString = ""

        //iterate array 裡所有的 element
        for dictionary in dataArray {

            //your code: 將dictionary轉成csv一筆資料的格式，更新finalString

        }

        //寫入data.txt檔案
        writeStringToFile(writeString: finalString, fileName:
"data.txt")

    }
```

(4) **完成載入 Dictionary Array 的 function**。我們需要將CSV格式的字串轉為Dictionary，此時，我們可以使用補充指令的 string 分割來解讀CSV的格式的字串：先用 separator "\n" 將每行分開成 array，也就是每個 array 的 element 為一筆資料。接著用 separator "," 將每筆資料的各個欄位分開。
我們可將讀取檔案並且將內容轉換成dictionary的功能包成一個 method ，請自行將此 method 補完。

```
func loadDataArray() {

    //宣告儲存Array最後的結果的變數
    var finalArray = [[String:Any]]()

    //讀取data.txt的檔案內容
    let csvString = readFileToString(fileName: "data.txt")

    //用"\n"將每一筆資料分開
    let lineOfString = csvString.components(separatedBy: "\n")

    //iterate 每一筆資料的string
    for line in lineOfString {


        //將這筆資料的string轉成dictionary的格式
        //your code here

        //將讀取出的這筆資料加入array
        //your code here
    }


    //將讀取出的finalArray取代掉原本的dataArray
    dataArray = finalArray

    //更新 tableview 與 介面資料
    tableView.reloadData()
    updateTotal()

}
```

(5) 最後，在適當的時機儲存、載入資料

※請自行完成

(6) 執行 app ，測試在停止，重新執行 app 後，上次修改的資料是否還存在

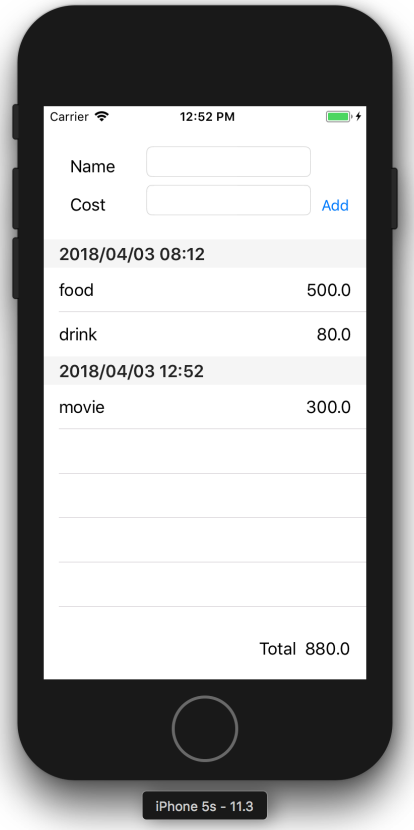
(7) 我們會在下次 Lab 中實做更完整的 Bookkeeping 功能

八. Demo

1. 請 Demo 最後的執行結果，**可以正確顯示Dictionary的資料**，總額也可以正確隨著更新。**重新打開 app 後，上次的資料還會存在。**

2. 加分題
   - **基礎加分題**：試著用自己的方法將每筆資料的時間資訊也顯示在介面上

   - **進階加分題**：使用 UITableView 的 section 區塊區分不同的**分鐘**，需顯示 section header title。
     1. 使用含有 section header title 的 tableview 來「**顯示**」資料
     2. 可以正常「**新增與刪除**」資料，可存檔，不用顯新增與刪除的示動畫



※進階加分題1,2,3 分開加分
※建議複製一份project來做
※~~基礎加分題做完才能做進階加分題~~
※超過下課時間後就不能Demo加分題

- **進階加分題提示**
  - 正常用法是用 section 區分不同的日期或是月份，但 Demo 會看不出來
  - 可參考 Dictionary 的 init(grouping:by:)
    https://developer.apple.com/documentation/swift/dictionary/2919592-init

```
//可自行去playground試試
let member:[[String:Any]] = [ ["name":"Abby"  ,"group":"A", "age":20],
                              ["name":"Betty" ,"group":"B", "age":32],
                              ["name":"Carl"  ,"group":"A", "age":18],
                              ["name":"Denny" ,"group":"B", "age":39],
                              ["name":"Emma"  ,"group":"C", "age":21],
                              ["name":"Frank" ,"group":"A", "age":36]]

let groupByNameResult = Dictionary.init(grouping: member) { (person:Dictionary<String,Any>) ->
String in
    return person["group"] as! String
}
print("\(groupByNameResult)")


let groupByAgeResult = Dictionary.init(grouping: member) { (person:Dictionary<String,Any>) ->
String in

    switch person["age"] as! Int {
    case 0...20:
        return "0~20"
    case 21...30:
        return "21~30"
    case 31...40:
        return "31~40"
    default:
        return "40~"
    }
}
print("\(groupByAgeResult)")
```

  - 遞增排序 array

```
let array = ["D","B","C","A"]
array.sorted() //["A", "B", "C", "D"]
```

  - 顯示正確的 section header 最少需要新增的 data source protocol function

```
//回傳有幾個section
func numberOfSections(in tableView: UITableView) -> Int

//回傳不同位置的section header要顯示什麼title
func tableView(_ tableView: UITableView, titleForHeaderInSection
section: Int) -> String?
```