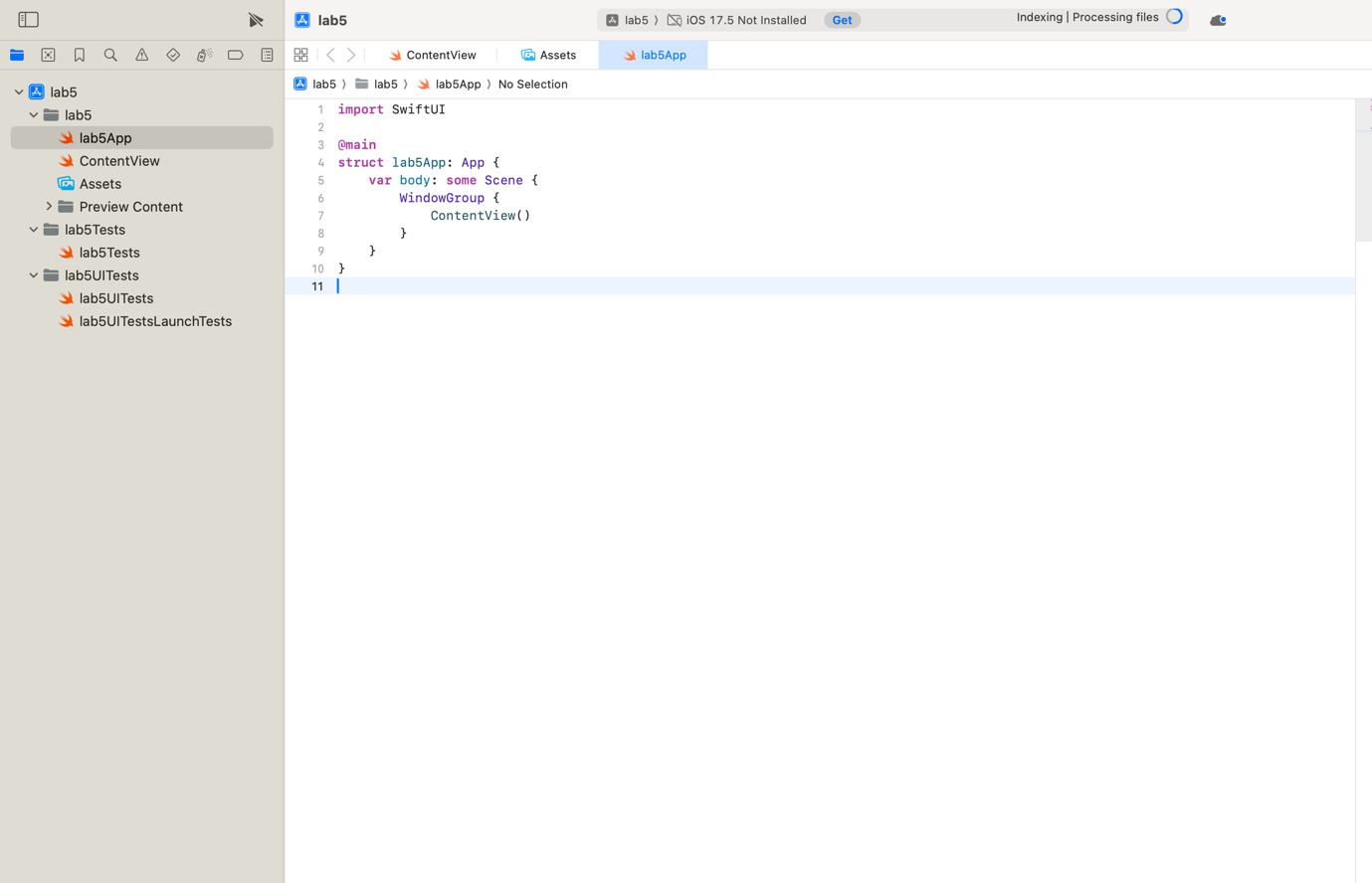
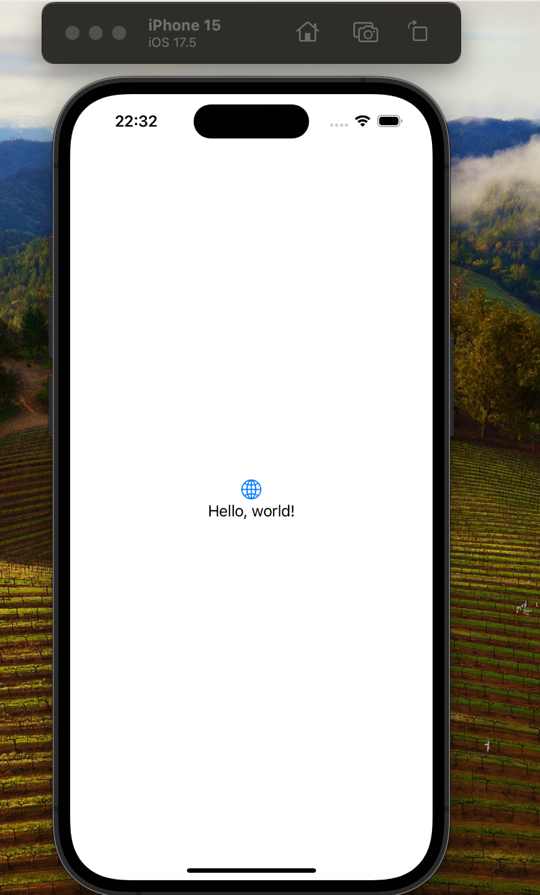
**Лабораторная работа #5**

Задание 1:





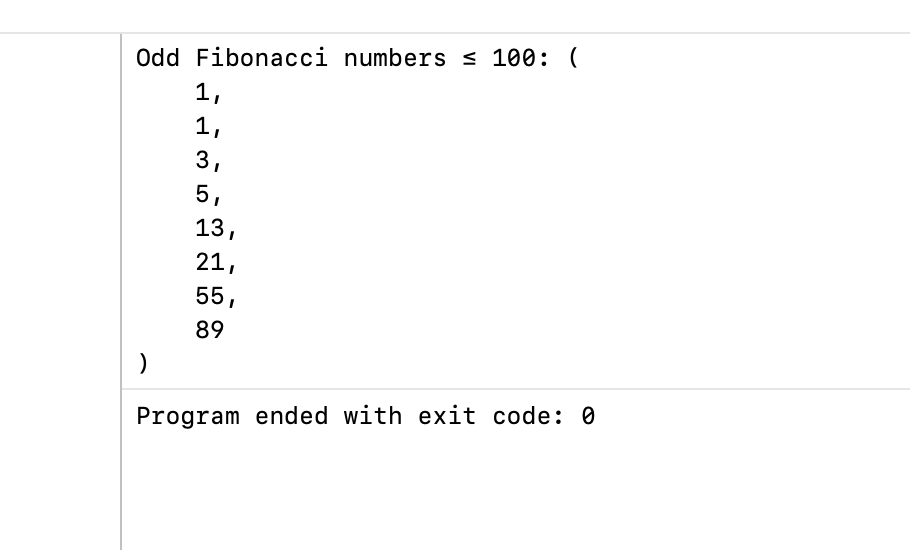
1. Какие способы запуска симулятора iOS возможны?

Через Xcode: выбрать схему с “Simulator” и нажать кнопку Run (▶).  
– Через меню Xcode: Xcode → Open Developer Tool → Simulator.  
– Из командной строки:  
• open -a Simulator.app  
• xcrun simctl boot <UDID> для загрузки конкретного устройства  
– Через скрипты и инструменты CI, вызывающие xcodebuild test или xcrun simctl автоматически.

2. Как управлять симулятором?

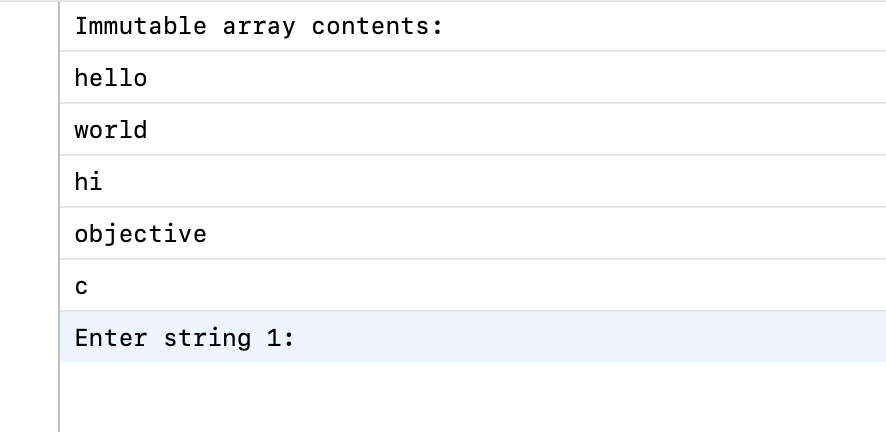
Меню Hardware в окне Simulator:  
• Rotate Left/Right (поворот устройства)  
• Home, Lock, Shake для эмуляции нажатий и встряски  
• Device → выбрать модель и версию iOS  
• Features → геолокация, клавиатура, звук, состояние сети  
– Горячие клавиши:  
• ⌘+←/⌘+→ — переключение между установленными версиями iOS  
• ⌘+⇧+H — эмуляция кнопки Home (один или два нажатия)  
• ⌘+I — отображение устройства в окне “Inspector”  
• ⌘+D — меню Debug приложения  
– Команды simctl:  
• xcrun simctl list — список устройств и рунтаймов  
• xcrun simctl boot <UDID> / shutdown <UDID> — запуск/выключение эмулятора  
• xcrun simctl install <UDID> <app> — установка приложения  
• xcrun simctl uninstall <UDID> <bundle-id> — удаление приложения  
• xcrun simctl launch <UDID> <bundle-id> — запуск установленного приложения  
• xcrun simctl erase <UDID> — сброс содержимого и настроек симулятора.

Задание 2:

  
  
 Расширения файлов:  
C language source file – .c  
C++ language source file – .cpp (или .cc)  
Header file – .h  
Objective-C source file – .m  
Objective-C++ source file – .mm  
Object (compiled) file – .o

 Способы вывода на экран в Objective-C:  
– NSLog(@"…");  
– printf("…\n");  
– fprintf(stdout, "…\n");  
– Для GUI-приложений через UIKit/AppKit: UILabel, UITextView, NSTextField и др.

Задание 3



**Протокол тестов**

Тест 1. printImmutableArray:

Вход: @[@"a",@"b"]

Ожидание: вывод строк "a" и "b" в консоль.

---

Тест 2. createEmptyMutableArray & countOfArray:

Ожидание: возвращает пустой массив, count = 0.

---

Тест 3. readStringsFromConsole (эмуляция ввода):

Вход: ввод "x","y"

Ожидание: массив @[@"x",@"y"], count = 2.

---

Тест 4. addElement & insertElement:

Изначально @[], после add @"a" → @[@"a"], после insert @"b" на 0 → @[@"b",@"a"].

---

Тест 5. removeLastElement & removeElementAtIndex & removeAll:

Для @[@"1",@"2",@"3"]:

removeLast → @[@"1",@"2"];

removeAtIndex0 → @[@"2"];

removeAll → @[].

**Ответы на вопросы**   
  
 **Какой метод используется для получения количества элементов в NSArray?**  
- (NSUInteger)count

 **Как правильно создать пустой mutable массив?**  
Варианты:  
A) NSArray \*array = [[NSArray alloc] init];  
B) NSMutableArray \*array = [[NSMutableArray alloc] init];  
C) NSArray \*array = [NSArray array];  
D) NSMutableArray \*array = [NSMutableArray array];  
**Правильно:** B и D.

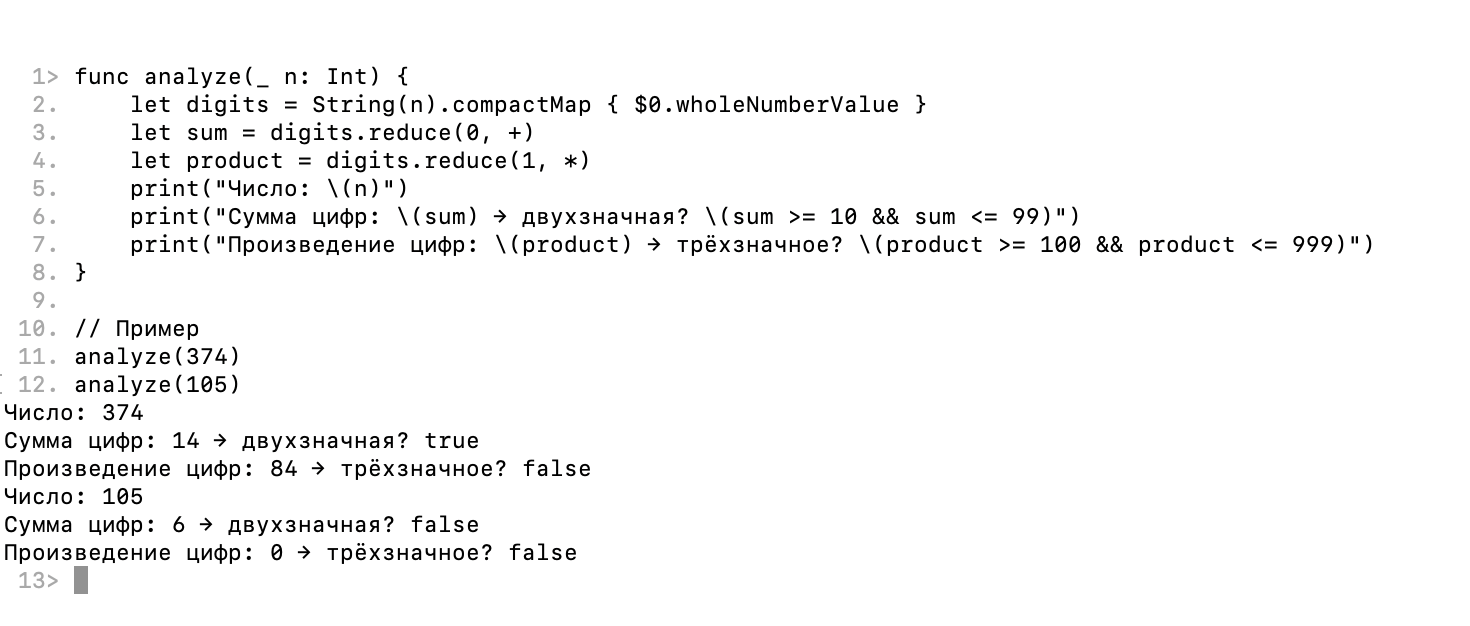
 **Какой метод используется для добавления элемента в конец mutable массива?**  
- (void)addObject:(ObjectType)anObject;

 **Что произойдет при попытке получить доступ к элементу массива по индексу, превышающему размер массива?**  
Во время выполнения приложение упадёт с исключением NSRangeException.

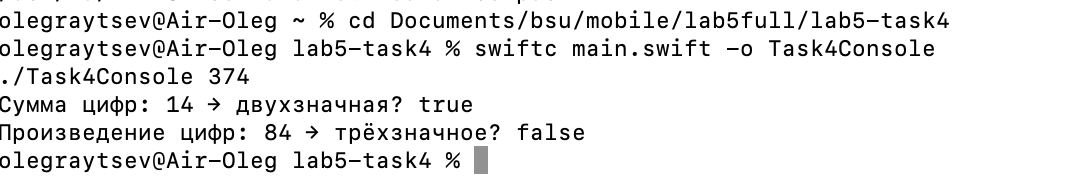
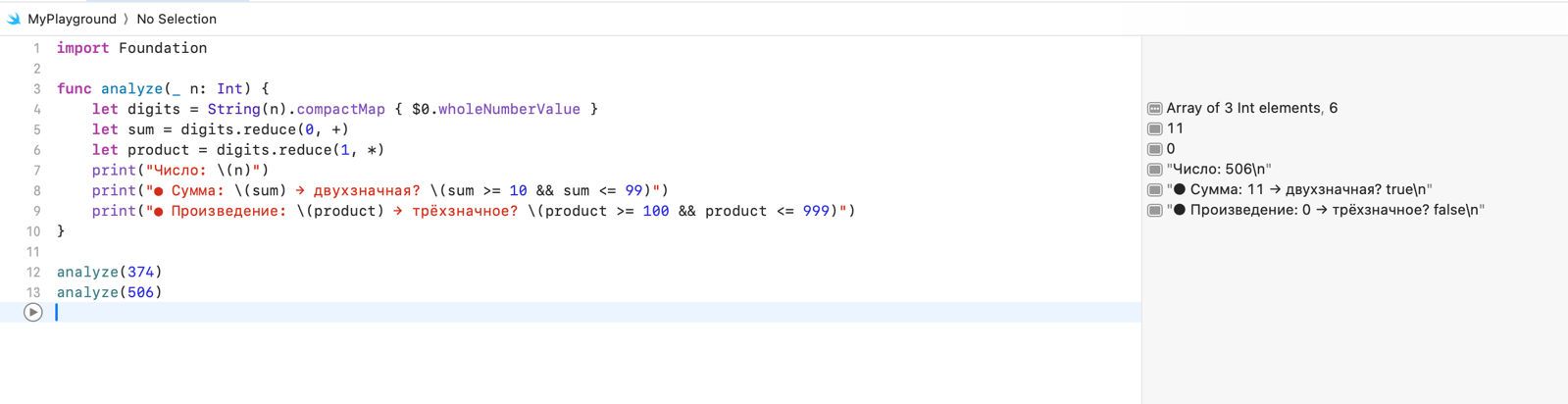
 **Как получить последний элемент массива?**  
array.lastObject или [array objectAtIndex:array.count - 1].

Задание 4

Swift REPL



SwiftC



Протоколы тестов:

Тест 1. n = 374

Ожидание:

Сумма = 3+7+4 = 14 → двухзначная? true

Произведение = 3\*7\*4 = 84 → трёхзначное? false

Тест 2. n = 105

Ожидание:

Сумма = 1+0+5 = 6 → двухзначная? false

Произведение = 1\*0\*5 = 0 → трёхзначное? false

Тест 3. n = 999

Ожидание:

Сумма = 27 → двухзначная? true

Произведение = 729 → трёхзначное? True

**Ответы на вопросы**

1. **Что такое Swift REPL и когда его используют?**  
   Swift REPL (Read–Eval–Print Loop) — интерактивная консоль для мгновенного запуска и отладки фрагментов Swift-кода без создания проекта. Полезен для быстрого прототипирования, экспериментов с API и изучения языка.
2. **Что такое Playground? Для каких задач?**  
   Playground в Xcode — интерактивная среда, где Swift-код компилируется «на лету» и отображает результат: тексты, графики и визуализации. Идеально подходит для учебных примеров, визуализации алгоритмов, UI-прототипов и экспериментов с API.
3. **Как компилировать Swift-приложение в консоли bash?**  
   Используйте компилятор swiftc:

bash

КопироватьРедактировать

swiftc <source1>.swift <source2>.swift … -o <executable>

Например:

bash

КопироватьРедактировать

swiftc main.swift -o Task4Console

Задание 5:

 **Как создать локальный git-репозиторий для проекта в Xcode?**  
– При создании нового проекта в диалоге “New Project” поставить галочку “Create Git repository on my Mac”.  
– Либо в любом существующем проекте через меню Source Control → New Git Repositories…, выбрать папку проекта и нажать Create.  
– Можно также в терминале перейти в корень проекта и выполнить git init, после чего Xcode автоматически подхватит репозиторий.

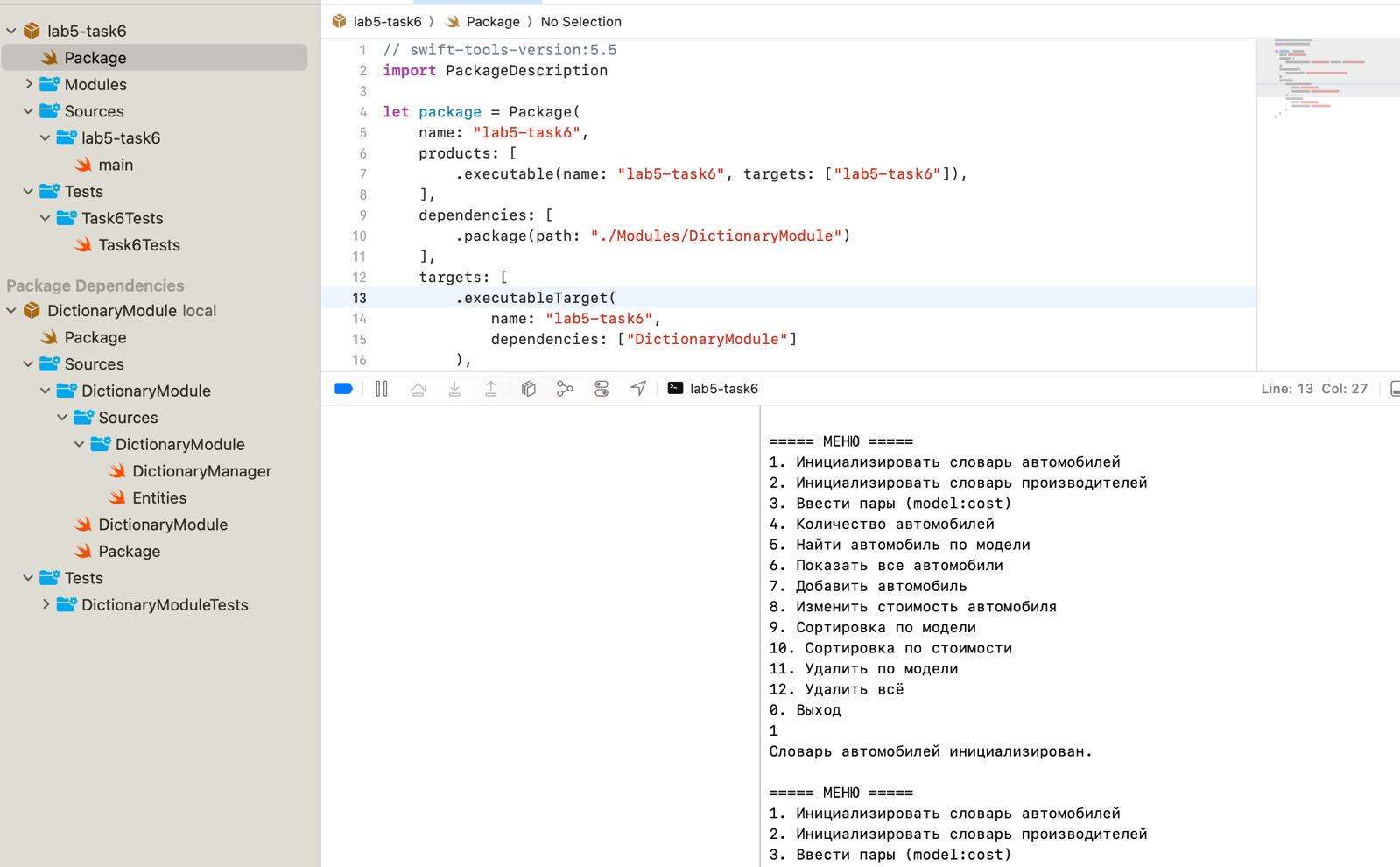
** Как добавить внешний (удалённый) репозиторий в Xcode?**  
– Через меню Source Control → Clone…, ввести URL удалённого репозитория и указать, куда его склонировать.  
– Открыть существующий локальный репозиторий, затем в Project Navigator щёлкнуть правой кнопкой по названию проекта → Source Control → Configure “YourProject” → Remotes → + → Add Remote…, задать имя (например origin) и URL.  
(Альтернативный способ: через терминал в папке проекта выполнить git remote add origin <URL>, и Xcode автоматически увидит новый удалённый.)

 **Как создать ветку репозитория в Xcode?**  
– Меню Source Control → New Branch…, задать имя ветки и от какой ветки ответвиться.  
– Или в Project Navigator выбрать репозиторий, правой кнопкой открыть Source Control → Branches → + → New Branch…, ввести имя ветки.

** Как отменить (откатить) последний commit в Xcode?**  
– Menu Source Control → Undo Last Commit.  
– Или открыть Source Control Navigator, выбрать ветку, найти нужный коммит, правой кнопкой → Revert Changes in Commit (или Cherry-Pick/Revert), что создаст обратный коммит.  
– Можно также в терминале выполнить git reset --soft HEAD~1 или git revert <commit-hash>, и изменения отобразятся в Xcode.

** Как слить (merge) ветки в Xcode?**  
– Menu Source Control → Merge Branch…, выбрать ветку-источник и нажать Merge into current branch.  
– В Source Control Navigator открыть список веток, правой кнопкой на ветке, которую надо влить → Merge Into Current Branch.  
– При возникновении конфликтов Xcode покажет конфликты в редакторе, где можно выбрать, как их разрешить.

Задание 6:



**5. Ответы на вопросы**

**1) Режимы работы среды xCode**

* **Редактирование (Editing)**: написание и правка кода.
* **Сборка (Building)**: компиляция и линковка проекта.
* **Отладка (Debugging)**: выполнение приложения с возможностью ставить точки останова, смотреть переменные, шагать по коду.
* **Тестирование (Testing)**: запуск модульных тестов и тестов производительности.
* **Archive/Distribution**: упаковка приложения для App Store или побочного распространения.

**2) Отличия Swift от C и C++**

* **Безопасность**: опционалы, автоматическое управление памятью (ARC), строгая типизация.
* **Синтаксис**: более лаконичный, встроенные строки Unicode, функциональные возможности (map/filter).
* **Модули**: встроенный пакетный менеджер Swift PM вместо заголовочных файлов.
* **Протоколы**: замена шаблонов C++ с «protocol-oriented programming».
* **Отсутствие пре- и пост- инкремента** (++/--), нет неинициализированных переменных.

**3) Способы отладки в xCode**

* Точки останова (breakpoints) и условные точки останова.
* LLDB-консоль для введения команд на лету.
* View Debugging для визуализации иерархий UI.
* Thread Sanitizer, Address Sanitizer, Undefined Behavior Sanitizer.
* Time Profiler и другие инструменты Instruments.

**4) Операторы Swift**

* Арифметические: +, -, \*, /, %.
* Сравнения: ==, !=, <, >, <=, >=.
* Логические: &&, ||, !.
* Присваивания: =, +=, -=, \*=, /=.
* Диапазоны: ... (closed), ..< (half-open).
* Тернарный: condition ? expr1 : expr2.
* Операторы объединения опционалов: ??.
* Оператор приведения типов: as, as?, as!.

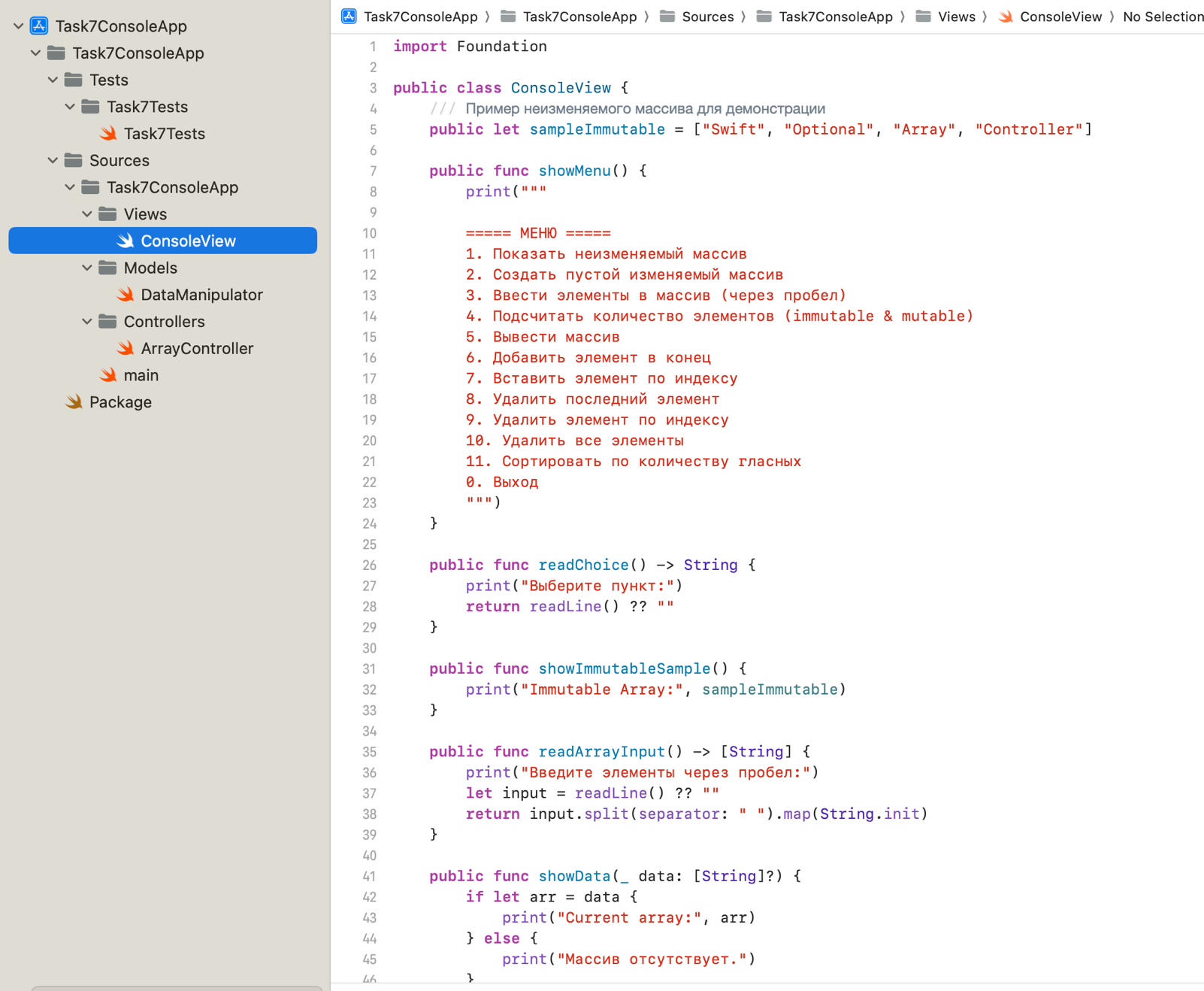
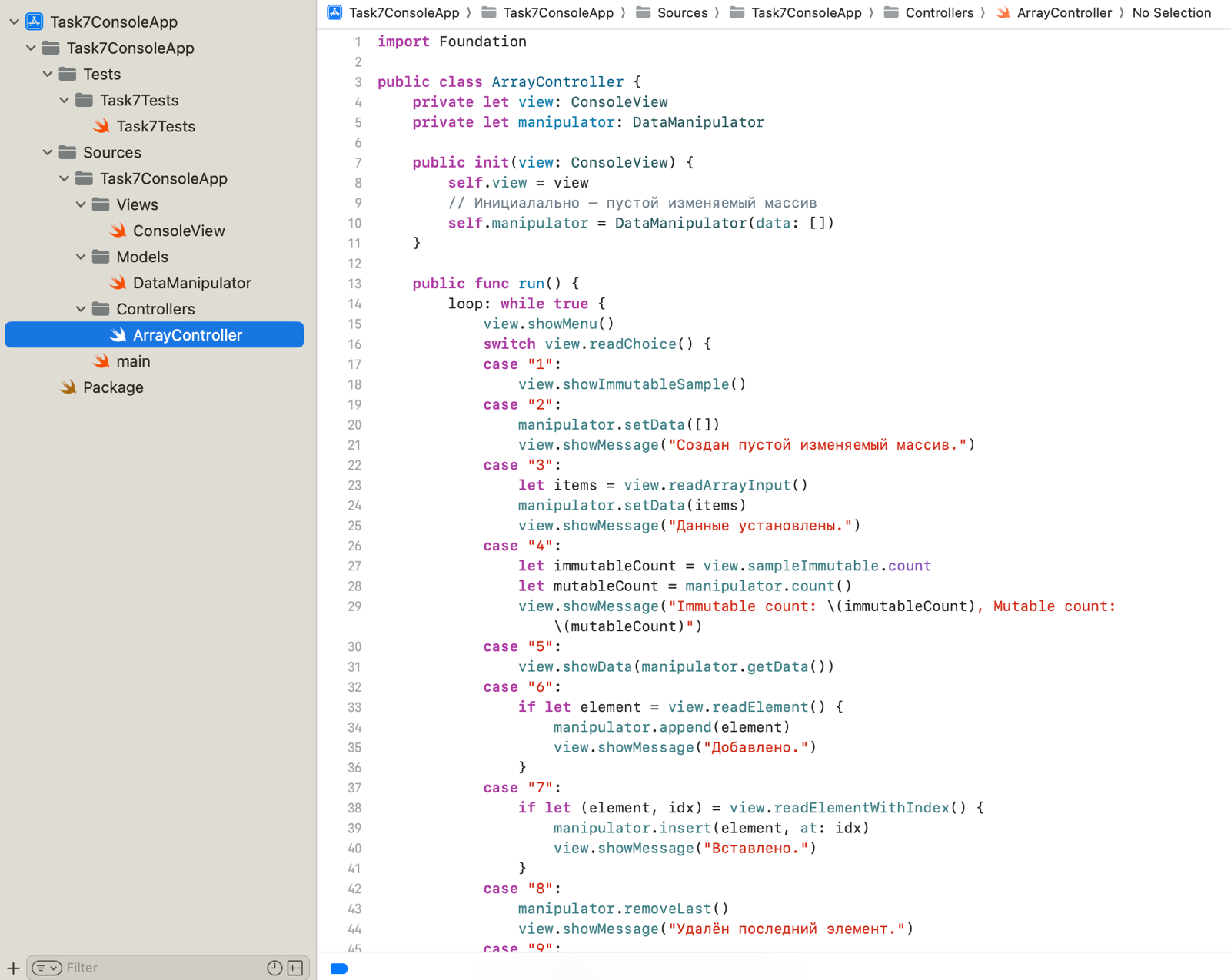
**5) Коллекции в Swift и их особенности**

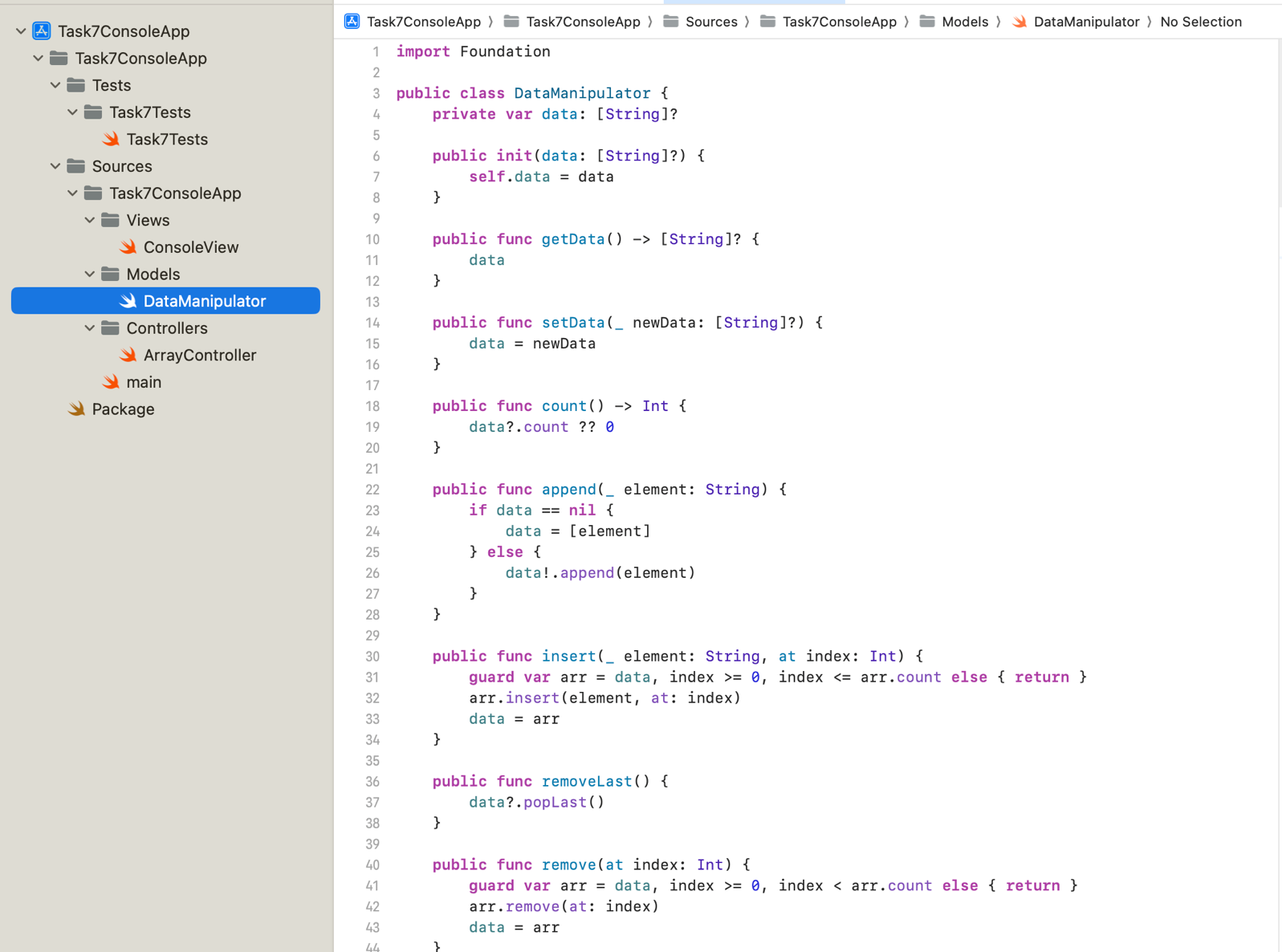
* **Array**: упорядоченный, индексируемый тип, копируется при изменении (copy-on-write).
* **Dictionary**: неупорядоченный набор пар „ключ–значение“, ключи уникальны, тоже copy-on-write.
* **Set**: неупорядоченный набор уникальных значений, хешируемые элементы.  
  Каждая коллекция — value type, обеспечивает безопасность потоков и оптимизацию копирований.

**6) Типы проектов в xCode**

* **App** (iOS/macOS/tvOS/watchOS) — графическое приложение.
* **Command Line Tool** — консольное приложение (наш вариант).
* **Framework/Library** — переиспользуемый модуль.
* **Playground** — интерактивная среда для экспериментов.
* **Unit Test Bundle/UI Test Bundle** — таргеты для тестирования.
* **Swift Package** — пакет Swift PM без UI.

Задание 7:





**1) Как создать пустой массив типа Int в Swift?**

swift

КопироватьРедактировать

var emptyArray: [Int] = []

// или явнее

var emptyArray2 = [Int]()

**2) Что произойдет, если объявить массив как константу (let) и попытаться изменить его размер?**  
При объявлении через let массив становится неизменяемым (immutable). Любая попытка добавить или удалить элементы приведёт к ошибке компиляции:

pgsql

КопироватьРедактировать

error: cannot use mutating member on immutable value: 'numbers' is a 'let' constant

**3) Какой метод используется для добавления элемента в конец массива?**

swift

КопироватьРедактировать

array.append(element)

Например:

swift

КопироватьРедактировать

var nums = [1, 2]

nums.append(3) // nums теперь [1, 2, 3]

**4) Какой тип данных должен быть у всех элементов массива?**  
Все элементы массива должны быть одного и того же типа — тем, который указан в его объявлении. В массиве [T] все элементы должны быть типа T.

**5) Что означает “массив является универсальной коллекцией”?**  
Это значит, что массив в Swift реализован как обобщённый (generic) тип Array<Element>, где Element может быть любым типом. Благодаря этому можно создавать массивы с любым типом элементов, сохраняя при этом типобезопасность.

**6) Какой метод используется для перебора элементов массива с получением индекса?**  
Для этого обычно используют метод enumerated(), который возвращает последовательность пар (index, element). Пример:

swift

КопироватьРедактировать

let fruits = ["🍎", "🍊", "🍌"]

for (index, fruit) in fruits.enumerated() {

print("\(index): \(fruit)")

}