Практическая работа № 7

Классы и структуры.

Цель работы:

1. Изучить создание и инициализацию классов и структур, научиться применять на практике.
2. Получить навыки использования классов.
3. Создать консольное приложение, в основе работы которого будут использоваться классы и структуры.

Краткие теоретические сведения

Классы

Классы в Swift являются строительными блоками гибких конструкций. Подобно константам, переменным и функциям пользователь может определять свойства и методы класса. Swift предоставляет нам функциональность, с помощью которой, при объявлении классов пользователям не нужно создавать интерфейсы или файлы реализации. Swift позволяет нам создавать классы в виде одного файла, и внешние интерфейсы будут создаваться по умолчанию после инициализации классов.

### Преимущества:

* Наследование передает свойства одного класса другому классу
* Приведение типов позволяет пользователю проверять тип класса во время выполнения
* Деинициализаторы заботятся об освобождении ресурсов памяти
* Подсчет ссылок позволяет экземпляру класса иметь более одной ссылки

### Общие характеристики классов и структур

* Свойства определены для хранения значений
* Индексы определены для обеспечения доступа к значениям
* Методы инициализируются для улучшения функциональности
* Начальное состояние определяется инициализаторами
* Функциональность расширена за пределы значений по умолчанию
* Подтверждение стандартов функциональности протокола

### Синтаксис

Classclassname {

Definition 1

Definition 2

---

Definition N

}

### Определение класса

classstudent {

varstudname: String

varmark: Int

var mark2: Int

}

Синтаксис для создания экземпляров

letstudrecord = student()

**Пример:**

*сlass MarksStruct{*

*var mark:Int*

*init(mark:Int){*

*self.mark= mark*

*}*

*}*

*classstudentMarks{*

*var mark =300*

*}*

*let marks =studentMarks()*

*print("Mark is \(marks.mark)")*

Когда мы запускаем вышеуказанную программу, используя площадку, мы получаем следующий результат –

*Markis 300*

### Доступ к свойствам класса в качестве ссылочных типов

Свойства класса могут быть доступны с помощью «.»синтаксис. Имя свойства отделяется знаком «.» после имени экземпляра.

*classMarksStruct{*

*var mark:Int*

*init(mark:Int){*

*self.mark= mark*

*}*

*}*

*classstudentMarks{*

*var mark1 =300*

*var mark2 =400*

*var mark3 =900*

*}*

*let marks =studentMarks()*

*print("Mark1 is \(marks.mark1)")*

*print("Mark2 is \(marks.mark2)")*

*print("Mark3 is \(marks.mark3)")*

Когда мы запускаем вышеуказанную программу, используя площадку, мы получаем следующий результат –

*Mark1 is 300*

*Mark2 is 400*

*Mark3 is 900*

Структуры

Swift предоставляет гибкие конструкции для использования структур. Применяя структуры, можно определить методы и свойства конструкций.

### В отличие от C и Objective C

* Структура не требует файлов реализации и интерфейса.
* Структура позволяет нам создавать один файл и автоматически расширять его интерфейс для других блоков.

В Структуре значения переменных копируются и передаются в последующих кодах путем возврата копии старых значений, чтобы значения не могли быть изменены.

Структуры описываются с помощью ключевого слова 'Struct'

### Синтаксис

struct nameStruct {

Definition 1

Definition 2

---

Definition N

}

## Определение структуры

Рассмотрим пример, предположим, что мы должны получить доступ к записям студентов, содержащим оценки трех предметов, и узнать три предмета. Здесь markStruct используется для инициализации структуры с тремя отметками как тип данных «Int».

structMarkStruct {

varmark1: Int

varmark2: Int

var mark3: Int

}

## Доступ к структуре и ее свойствам

Члены структуры доступны по имени структуры. Экземпляры структуры инициализируются ключевым словом let.

*struct studentMarks{*

*var mark1 =100*

*var mark2 =200*

*var mark3 =300*

*}*

*let marks =studentMarks()*

*print("Mark1 is \(marks.mark1)")*

*print("Mark2 is \(marks.mark2)")*

*print("Mark3 is \(marks.mark3)")*

Когда мы запускаем вышеуказанную программу, используя площадку, мы получаем следующий результат –

*Mark1 is 100*

*Mark2 is 200*

*Mark3 is 300*

Доступ к оценкам учащихся осуществляется по структуре «studentMarks». Элементы структуры инициализируются как mark1, mark2, mark3 со значениями целочисленного типа. Затем структура studentMarks () передается метке «mark» с ключевым словом «let». В дальнейшем «отметки» будут содержать значения элементов структуры. Теперь значения печатаются путем доступа к значениям элементов структуры с помощью «.» сегоинициализированнымиименами.

*struct MarksStruct{*

*var mark:Int*

*init(mark:Int){*

*self.mark=mark*

*}*

*}*

*var aStruct=MarksStruct(mark:98)*

*var bStruct=aStruct// aStruct and bStruct are two structs with the same value!*

*bStruct.mark=97*

*print(aStruct.mark)// 98*

*print(bStruct.mark)// 97*

Когда мы запускаем вышеуказанную программу, используя площадку, мы получаем следующий результат –

*98*

*97*

## Лучшие практики использования конструкций

Язык Swift обеспечивает функциональность для определения структур как пользовательских типов данных для построения функциональных блоков. Экземпляры структуры передаются по значению в определенные блоки для дальнейших манипуляций.

### Нужно иметь структуры

* Инкапсулировать простые значения данных.
* Копировать инкапсулированные данные и связанные с ними свойства по «значениям», а не «ссылкам».
* Структура для «Копировать» и «Ссылка».

Структуры в Swift передают свои члены со своими значениями, а не ссылки.

*Struct markStruct{*

*var mark1:Int*

*var mark2:Int*

*var mark3:Int*

*init(mark1:Int, mark2:Int, mark3:Int){*

*self.mark1 = mark1*

*self.mark2 = mark2*

*self.mark3 = mark3*

*}*

*}*

*varmarks=markStruct(mark1:98, mark2:96, mark3:100)*

*print(marks.mark1)*

*print(marks.mark2)*

*print(marks.mark3)*

Когдамызапускаемвышеуказаннуюпрограмму, используяплощадку, мыполучаемследующийрезультат –

*98*

*96*

*100*

### Другойпример

*Struct markStruct{*

*var mark1:Int*

*var mark2:Int*

*var mark3:Int*

*init(mark1:Int, mark2:Int, mark3:Int){*

*self.mark1 = mark1*

*self.mark2 = mark2*

*self.mark3 = mark3*

*}*

*}*

*Var fail=markStruct(mark1:34, mark2:42, mark3:13)*

*print(fail.mark1)*

*print(fail.mark2)*

*print(fail.mark3)*

Когдамызапускаемвышеуказаннуюпрограмму, используяплощадку, мыполучаемследующийрезультат –

*34*

*42*

*13*

Задание для самостоятельного выполнения

1. Создать, инициализировать и вывести на экран класс в соответствии с вариантом задания
2. Создать, инициализировать и вывести на экран структуру в соответствии с вариантом задания
3. Описать структуры на языке Swift, инициализировать поля, создать массив значений, ввести значение для поиска и вывести прочие поля найденного элемента значения

Варианты

1. Автобус (номер маршрута, название начальной и конечной остановки, количество остановок по маршруту) Ввести номер маршрута
2. Книга (название книги, автор, издательство, количество страниц) Ввести название
3. Файл (имя, время создания, дата создания, размер) Ввести имя
4. Спортсмен (ФИО, вид спорта, сколько лет занимается) Ввести ФИО
5. Студент (ФИО, группа, средний балл зачетки) Ввести ФИО
6. Фирма (название, вид услуг, годовой оборот) Ввести название
7. Компьютер (модель, производитель, частота процессора, количество ядер, объем памяти) Ввести модель
8. Спорт (вид спорта, количество занимающихся в городе, средняя стоимость комплекта инвентаря) Ввести вид спорта
9. Автомобиль (марка, фирма-производитель,год выпуска, пробег) Ввести марку
10. Статья (название, автор, название журнала, год и месяц издания) Ввести название и автора