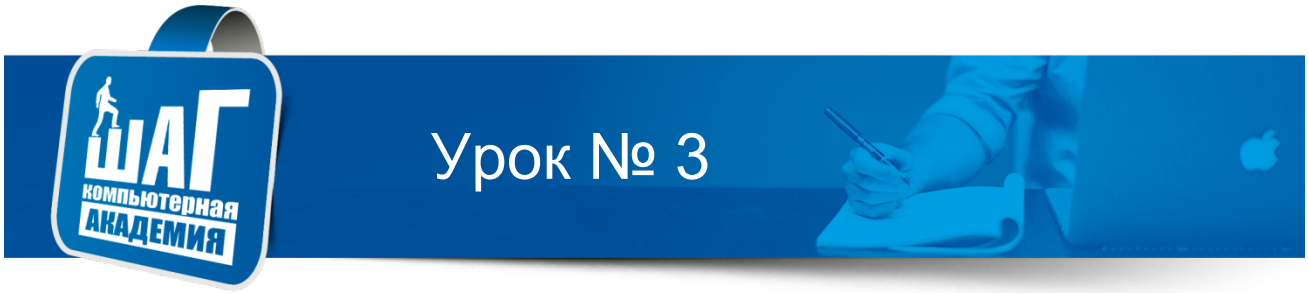

Курс: «Разработка приложений под мобильные устройства на основе iOS»

Тема: Условия. Циклы. Массивы

План

1. Условия выбора **if else**
2. Циклы **for**, **for in**
3. Циклы **while**, **do while**
4. Массивы **NSArray**
5. Словари **NSDictionary**

Резюме

1. Условия выбора IF-ELSE

```
if ("какое-то условие")
{
        /* Этот код выполнится, если условие даст YES. */
}
else //else - это ключевое слово, обеспечивающее выполнение кода в случае невыполнения условия.
{
        /* Этот код выполнится, если условие даст NO. */
}
```

Пример сравнения простых чисел

```
// age это целочисленная переменная, хранящая возраст пользователя
if (age > 30)
{
    NSLog(@"Старше 30 лет.");
}
else
{
    NSLog(@"Моложе 30 лет.");
}
```

Операторы сравнения числовых значений:

== равенство

> больше чем

< меньше чем

>= больше или равно

!= не равно

Пример сравнения объектов

```
NSObject *object1 = [[NSObject alloc] init];
NSObject *object2 = object1;

if ([object1 isEqual:object2])
{
    NSLog(@"Объекты идентичны.");
}
else
{
    NSLog(@"Объекты различны.");
}
```

Если должны выполняться два и более условия, используется логический оператор И, представляемый двумя символами амперсанда **&&**.

Если должно выполняться хотя бы одно условие, используйте логический оператор ИЛИ, представляемый двумя символами вертикальной черты ||.

```
if ( (age >= 18) && (age < 65) )
{
    NSLog(@"Скорее всего приходиться работать.");
}
```

Кроме того, можно объединять условные выражения. Нужно всего лишь заключить одно выражение в фигурные скобки другого. Сначала будет проверено условие снаружи, а затем, если оно верно, следующее выражение внутри, и так далее:

```
if (age >= 18)
{
    if (age < 65)
    {
        NSLog(@"Скорее всего приходиться работать.");
    }
}
```

Сокращенная конструкция условного оператора:

```
//Полная
if (age > 30)
{
    str = @"Старше 30 лет.";
}
else
{
    str = @"Моложе 30 лет.";
}

//Сокращенная
str = age > 30 ? @"Старше 30 лет." : @"Моложе 30 лет.";
```

2. Циклы for, for in

for ("код, который должен выполниться перед началом цикла";
"условие, при котором будет завершен весь цикл";
"код, который выполнится во время итерации цикла")
{
// тело цикла
}

```
char *myString = "This is my string";
NSUInteger counter = 0;

for (counter = 0; counter < strlen(myString); counter++)
{
    char character = myString[counter];
    NSLog(@"%c", character);
}
```

for in - цикл перебора, перебирает составные объекты, к примеру, массивы, словари.

```
NSArray* stringsArray = [NSArray arrayWithObjects:@"one", @"two", @"three", nil];

for (NSString* string in stringsArray)
{
    NSLog(@"%@",string);
}
```

оператор **break** - оператор прерывание цикла
оператор **continue** - выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла

3. Циклы while, do while

while ("какое-то условие")
{
"Ваш код"
}

Конструкция **while () {}** по сути, идентична циклу **for**. Все начинается с проверки условия окончания. Если результат ложный, выражения внутри цикла не выполняются.

```
NSUInteger counter = 0;
while (counter < 10)
{
    NSLog(@"Counter = %lu", (unsigned long)counter);
    counter++;
}
```

Пример бесконечного цикла

```
while (YES)
{
    /* Бесконечный цикл. */
}
```

do
{
"Ваш код"
}
while ("какое-то условие")

В конструкции **do {} while ()**, команды между фигурными скобками будут выполнены как минимум **1** раз.

```
NSUInteger counter = 1;
do
{
    NSLog(@"Это мой любимый фильм.");
    counter = counter + 1;
}
while (counter <= 10);
```

4. Массивы NSArray

Массив — это упорядоченный набор объектов. Массивы позволяют индексировать доступ к их содержимому. Чаще всего (но не обязательно) элементы массива имеют один определенный тип.

Инициализация массива (создание массива состоящего из 3-х строк)

```
NSArray* stringsArray = [NSArray arrayWithObjects:@"one", @"two", @"three", nil];
```

эквивалентная запись

```
NSArray* stringsArray = @[@"one", @"two", @"three"];
```

Доступ к элементам массива

Получение объекта по индексу (номеру)

```
NSString* string = [stringsArray objectAtIndex:0];
```

эквивалентная запись

```
NSString* string = stringsArray[0];
```

Получение первого и последнего элемента массива

```
NSString* firstString = [stringsArray firstObject];
NSString* lastString = [stringsArray lastObject];
```

5. Словари NSDictionary

NSDictionary. Словари управляют парами ключей и значений. Пара ключ-значение в словаре называется записью. Каждая запись состоит из одного объекта, представляющего ключ, а второй объект, который является значением ключа. В словаре, ключи являются уникальными, то есть, в одном словаре нет двух эквивалентных ключей.

Инициализация словаря (создание словаря состоящего из 2-х записей)

```
NSDictionary *dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
                           @"object1", @"key1",
                           @"object2",@"key2",
                           nil];
```

эквивалентная запись инициализации

```
NSDictionary *dictionary = @{@"key1":@"object1", @"key2":@"object2"};
NSLog(@"\ndictionary %@",dictionary);
```

результат выполнения кода

```
dictionary {
    key1 = object1;
    key2 = object2;
}
```

Доступ к элементам словаря

```
id object = [dictionary objectForKey:@"key1"];
```

эквивалентная запись

```
id object = dictionary[@"key1"];
```