

Записать Логические значения

Логические величины принимают всего два возможных значения: `true` или `false`. Они особенно удобны в условных командах, в которых выполнение блока кода зависит от того, истинно или ложно некоторое условие.



Записать Математические операции и сравнения

При помощи операторов $<$ и $>$ можно сравнить два значения и проверить, какое из них больше другого.

Оператор $==$ (два знака равенства) проверяет, что два значения равны, а оператор $!=$ проверяет, что два значения не равны,

$<=$ проверяет, что второе значение меньше или равно первому, а оператор $>=$ проверяет, что второе значение больше или равно первому.

Результатом сравнения является логическое значение (true или false).

Математические операции и сравнения

$4 < 6$	true	$4 > 6$	false
$2+2 == 5$	false	$2+2 != 5$	true
$4 \leq 6$	true	$4 \geq 4$	true

Вопросы

Какая функция из пакета `fmt` используется для вывода текста на консоль?

- `Fscan`
- `Scan`
- `Sprintf`
- `Println`

Вопросы

Какая команда используется в Go для импорта пакетов?

- Include
- Import
- Require
- load

Вопросы

Вставьте недостающую часть приведенного ниже кода, чтобы вывести "Привет, Мир!".

```
package main

import ("fmt")

func main() {
            ("Привет, Мир!")
}
```



Вопросы

Напишите программу, которая вычисляет и
выводит сумму $99+153$



Вопросы

Напишите программу, при помощи которой вы найдете символ соответствующий числовому коду 46

*Подсказка: используйте руны



Вопросы

Даны два числа 15897 и 1731.
Вычислить их сумму, разность,
произведение и частное.

Типы

```
package main

import (
    "fmt"
    "math"
    "strings"
)
```

```
func main() {
    fmt.Println(math.Floor("head first go"))
    fmt.Println(strings.Title(2.75))
}
```

*Обычно получает число
с плавающей точкой!*



Обычно получает строку!



Типы

```
package main
```

```
import (  
    "fmt"  
    "math"  
    "strings"  
)
```

```
func main() {  
    fmt.Println(math.Floor("head first go"))  
    fmt.Println(strings.Title(2.75))  
}
```

Обычно получает число
с плавающей точкой!

Обычно получает строку!

Ошибки.

```
cannot use "head first go" (type string) as type float64 in argument to math.Floor  
cannot use 2.75 (type float64) as type string in argument to strings.Title
```

Записать Типы

Значения в Go делятся на разные **типы**, которые определяют, для чего они могут использоваться.

В языке Go используется **статическая типизация** — это означает, что типы всех значений известны еще до запуска программы. Функции ожидают, что их аргументы относятся к конкретным типам, а их возвращаемые значения тоже имеют типы (которые могут совпадать или не совпадать с типами аргументов). Если случайно использовать неправильный тип значения в неподходящем месте, компилятор Go выдаст сообщение об ошибке. И это очень хорошо: вы узнаете о существовании проблемы до того, как о ней узнают пользователи!

Записать Типы

Чтобы узнать тип любого значения, передайте его функции `TypeOf` из пакета `reflect`. Давайте узнаем типы некоторых значений, которые уже встречались в примерах программ:

```
package main
```

```
import (
```

```
    "fmt"
```

```
    "reflect"
```

```
)
```

```
func main() {
```

```
    fmt.Println(reflect.TypeOf(42))
```

```
    fmt.Println(reflect.TypeOf(3.1415))
```

```
    fmt.Println(reflect.TypeOf(true))
```

```
    fmt.Println(reflect.TypeOf("Hello, Go!"))
```

```
}
```

Импортируем пакет
«reflect», чтобы использо-
вать его функцию `TypeOf`.

Возвращает тип
своего аргумента.

Результат.

```
int  
float64  
bool  
string
```