



Преговор

1. Каква ще е стойността на променливата **a** след изпълнението на следната програма:

```
int a = 5;  
switch (a) {  
    case 5:  
    case 6:  
        a = a + 1;  
        break;  
    default:  
        a = a + 2;  
        break;  
}
```

0

5

6

7

1. Каква ще е стойността на променливата **a** след изпълнението на следната програма:

```
int a = 5;  
switch (a) {  
    case 5:  
    case 6:  
        a = a + 1;  
        break;  
    default:  
        a = a + 2;  
        break;  
}
```

0

5

6

7

2. Какво ще се отпечата на конзолата, ако изпълним следната команда:

```
console.log(!(5 === 5) && (4 + 1 === 5));
```

True

False

Runtime
error

Compile time
error

2. Какво ще се отпечата на конзолата, ако изпълним следната команда:

```
console.log(!(5 === 5) && (4 + 1 === 5));
```

True

False

Runtime
error

Compile time
error

3. Какво ще се отпечата на конзолата, ако изпълним следната команда:

```
console.log(!(3 === 3) || (3 === 5));
```

True

Runtime
error

False

Compile time
error

3. Какво ще се отпечата на конзолата, ако изпълним следната команда:

```
console.log(!(3 === 3) || (3 === 5));
```

True

Runtime
error

False

Compile time
error

4. Какво ще се отпечата на конзолата, ако изпълним следната проверка:

```
console.log(!(3 > 5) || (1 === 1));
```

Compile time
error

Runtime
error

False

True

4. Какво ще се отпечата на конзолата, ако изпълним следната проверка:

```
console.log(!(3 > 5) || (1 === 1));
```

Compile time
error

Runtime
error

False

True

5. Какво ще се отпечата на конзолата, ако изпълним следната логическа проверка:

```
let number = 101;  
if (number >= 1)  
    console.log("Larger than 1");  
if (number <= 101)  
    console.log("Less than 101");  
console.log("Equal to 101");
```

Larger than 1

Less than 101

Equal to 101

Compile
time error

5. Какво ще се отпечата на конзолата, ако изпълним следната логическа проверка:

```
let number = 101;  
if (number >= 1)  
    console.log("Larger than 1");  
if (number <= 101)  
    console.log("Less than 101");  
console.log("Equal to 101");
```

Larger than 1

Less than 101

Equal to 101

Compile
time error

6. Какво ще се отпечата на конзолата, ако изпълним следната логическа проверка:

```
let role = "Administrator";  
let password = "SoftUni";  
if(role === "SoftUni") {  
    if(password === "SoftUni") {  
        console.log("Welcome!");  
    }  
}
```

"Welcome!"

Runtime error

No output

Compile time
error

6. Какво ще се отпечата на конзолата, ако изпълним следната логическа проверка:

```
let role = "Administrator";  
let password = "SoftUni";  
if(role === "SoftUni") {  
    if(password === "SoftUni") {  
        console.log("Welcome!");  
    }  
}
```

"Welcome!"

Runtime error

No output

Compile time
error

Повторения (цикли)

Прости повторения с `while`-цикъл



СофтУни

Преподавателски екип



Software
University



SoftUni
Foundation



Софтуерен университет

<http://softuni.bg>

Имате въпроси?

sli.do

#pb-nov



**Увеличаване и намаляване на стойността
на променливи**

- Инкрементиране - увеличаването на стойността на дадена променлива
 - Извършва се чрез оператори за инкрементиране - префиксни и постфиксни

Пример	Име	Резултат
++a	Пре-инкрементация	Увеличава стойността с единица и връща a
a++	Пост-инкрементация	Връща a и увеличава стойността с единица

- Извършва се само върху променливи, които имат числена стойност

- Пре-инкрементация

```
let a = 1;  
console.log(++a);  
console.log(a);
```

Стойността на променливата a се увеличава с 1 и след това се принтира

```
// 2  
// 2
```

- Пост-инкрементация

```
let a = 1;  
console.log(a++);  
console.log(a);
```

Първо се принтира променливата a и след това се увеличава с 1

```
// 1  
// 2
```

- Декрементиране – намаляването на стойността на дадена променлива
 - Извършва се чрез оператори за декрементиране – префиксни и постфиксни

Пример	Име	Резултат
--a	Пре-декрементация	Намалява стойността с единица и връща a
a--	Пост-декрементация	Връща a и намалява стойността с единица

- Извършва се само върху променливи, които имат числена стойност

- Пре-декрементация

```
let a = 1;  
console.log(--a); // 0  
console.log(a);  // 0
```

Стойността на променливата a се намалява с 1 и след това се принтира

- Пост-декрементация

```
let a = 1;  
console.log(a--); // 1  
console.log(a);  // 0
```

Първо се принтира променливата a и след това се намалява с 1

A background network diagram consisting of a grid of light gray lines intersecting at various points. At these intersections, there are several circles of different sizes, some solid light gray and some hollow, creating a web-like structure.

while

while-цикъл

Повторение докато е вярно дадено условие

Повторения (цикли) – while-цикъл

- В програмирането често се налага да изпълним блок с команди няколко пъти
 - За целта използваме цикли – **while**, **for** и други

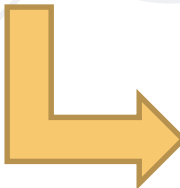


while-цикъл – пример

```
let a = 5;
```

Условие за прекратяване
на повторението

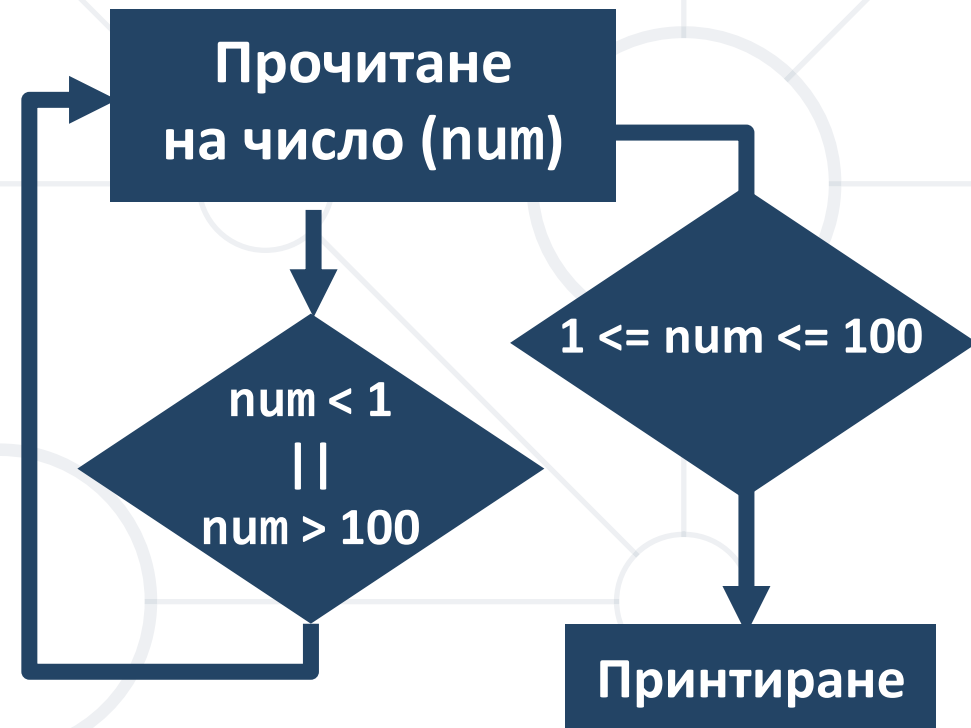
```
while (a <= 10) {  
    console.log("a = " + a);  
    a++;  
}
```



```
a = 5  
a = 6  
a = 7  
a = 8  
a = 9  
a = 10
```

Число в диапазона [1...100] - условие

- Напишете програма, която:
 - Прочита цяло число
 - Проверява дали е в диапазона [1...100]
- При:
 - Невалидно число, прочита ново
 - Намиране на число в диапазона, прекратява изпълнение



Число в диапазона [1...100] - решение

```
function numbersInRange(input) {  
  let number = Number(input.shift());  
  
  while (number < 1 || number > 100) {  
    number = Number(input.shift());  
  }  
  console.log(number);  
}
```

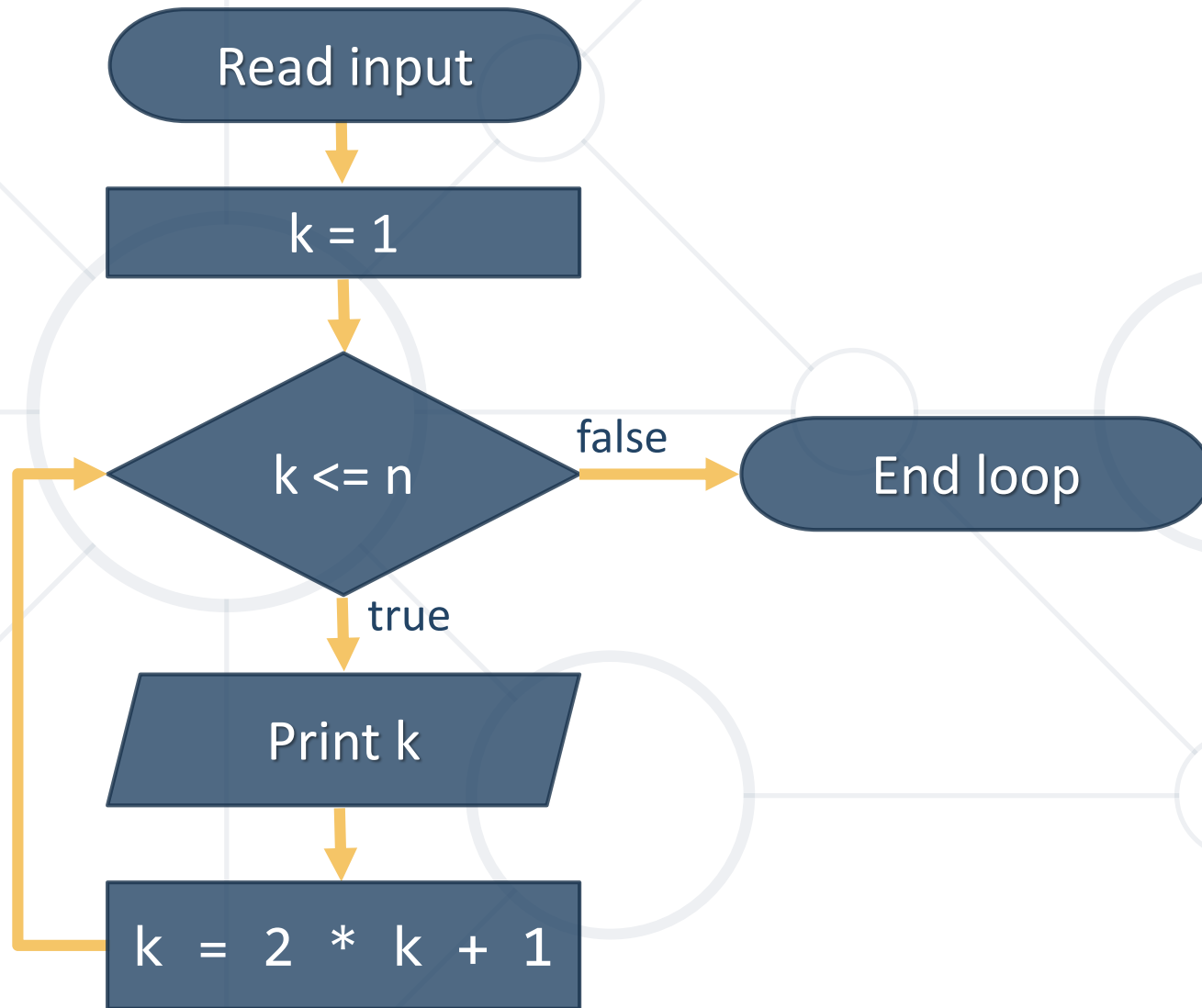
- Напишете програма, която:
 - Първоначално прочита потребителско име и парола на потребителски профил
 - Прочита парола за вход и проверява дали е коректна
 - При:
 - Невалидна парола, прочита нова
 - При коректно въведена парола, прекратява изпълнение

```
function password(input) {  
    let username = input.shift();  
    let password = input.shift();  
  
    let input = input.shift();  
    while (input !== password) {  
        input = input.shift();  
    }  
  
    console.log(`Welcome: ${username}!`);  
}
```

Редица числа $2k+1$ - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Отпечатва всички числа $\leq n$ от редицата: 1, 3, 7, 15, 31, ...
 - Всяко следващо число е равно на **предишното** $* 2 + 1$

1, $(1 * 2) + 1 =$ **3**, $(3 * 2) + 1 =$ **7**, $(7 * 2) + 1 =$ **15** ...



Редица числа $2k+1$ - решение

```
function sequence(input) {  
  let number = Number(input.shift());  
  let k = 1;  
  while (k <= number) {  
    console.log(k);  
    k = k * 2 + 1;  
  }  
}
```

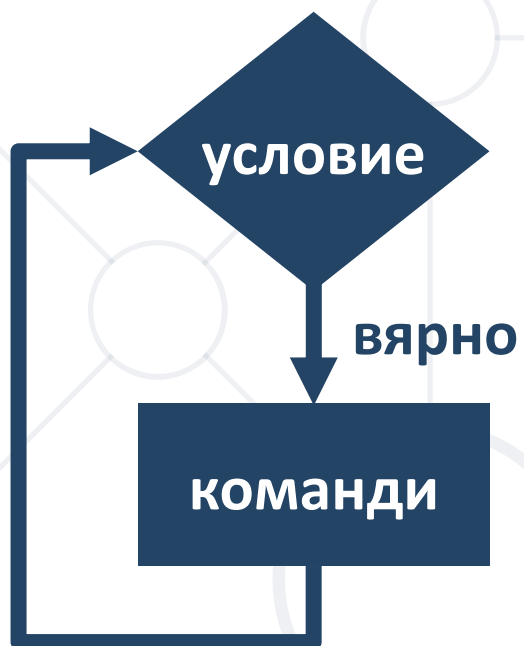
Повторение докато е в
сила условието $k \leq n$



Прекъсване чрез оператор `break`

Безкрайни цикли

- Безкраен цикъл – повтаряне на блок от код безкраен брой пъти:



Условието е винаги
вярно

```
while (true) {  
    console.log("Infinite loop");  
}
```



- Оператор **break** – прекъсва цикъла

```
while (true) {  
    console.log("Infinite loop");  
    if (...) {  
        break;  
    }  
}
```

Условие за прекъсване на
цикъла

- Напишете програма, която:
 - Чете n – на брой числа, които представляват вноски по банкова сметка
 - При всяка вноска принтира:
"Increase: {сумата}"



- Ако се въведе отрицателно число да се изпише **"Invalid operation!"** и програмата да приключи
- Накрая на програмата трябва да се изпише:
"Total: {общата сума в сметката}"



Баланс на сметка - условие(3)

- Примерен вход и изход:

3
5.51
69.42
100



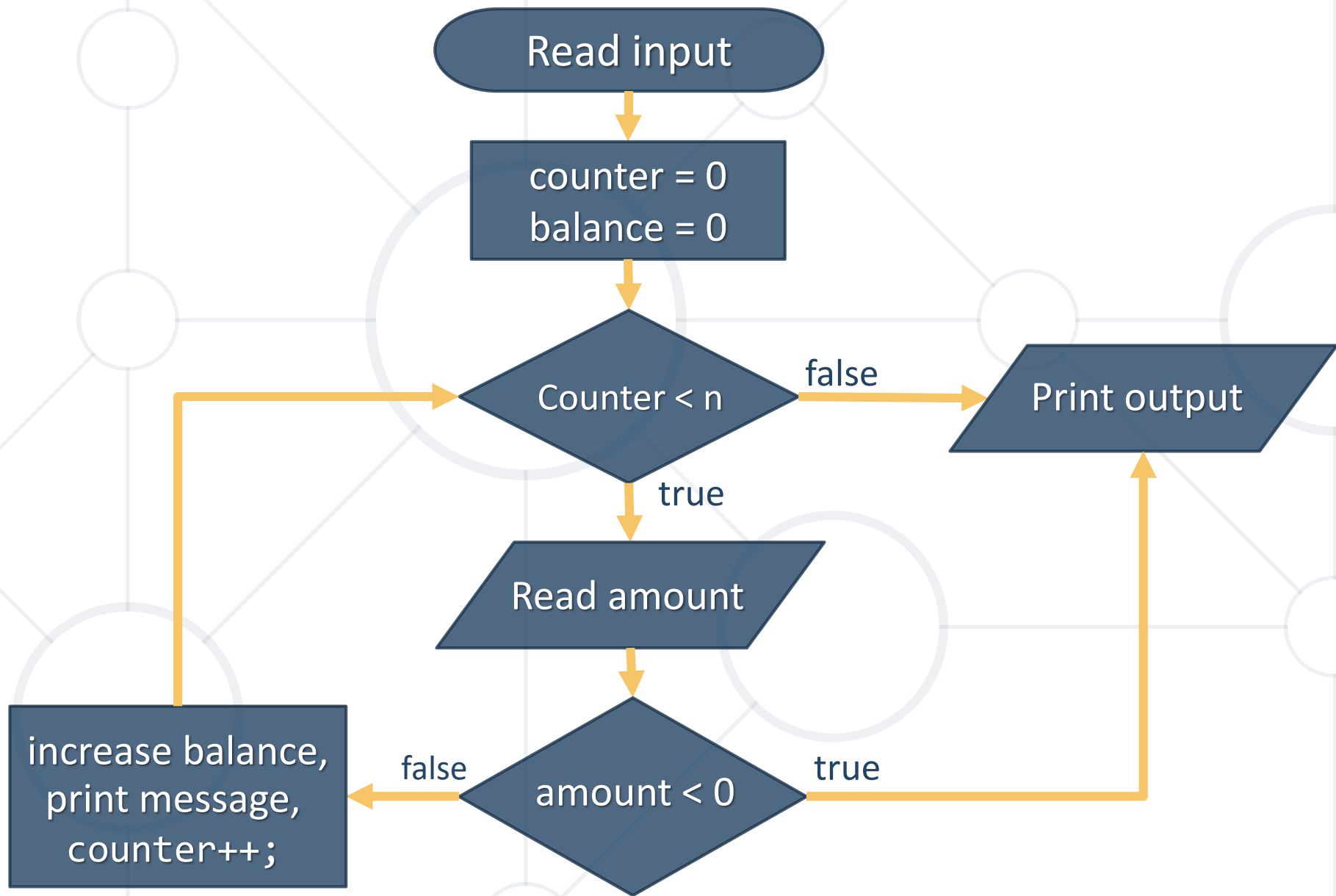
Increase: 5.51
Increase: 69.42
Increase: 100.00
Total: 174.93

5
120
45.55
-150



Increase: 120.00
Increase: 45.55
Invalid operation!
Total: 165.55





```
function accountBalance(input) {  
  let n = Number(input.shift());  
  let counter = 0;  
  let balance = 0.0;  
  while (counter < n) {  
    let amount = Number(input.shift());  
    if (amount < 0) { //TODO: Print message and exit the loop}  
      balance += amount;  
      console.log(`Increase: ${amount.toFixed(2)}`);  
      counter++;  
    }  
    console.log(`Total: ${balance.toFixed(2)}`);  
  }  
}
```

Най-голямо число - пример

- Напишете програма, която:
 - Получава число(*n*) от потребителя
 - Взема числа *n* последователни пъти
 - Намира най-голямото измежду тях
- Примерен вход и изход:

2
100
99



100

3
-10
20
-30



20

4
45
-20
7
99



99

5
3

Най-голямо число - решение

```
let n = Number(input[0]);
let counter = 0;
let max = Number.MIN_SAFE_INTEGER;
while (counter < n) {
  let num = Number(input.shift());
  counter++;
  if (num > max) {
    max = num;
  }
}
console.log(max);
```


Най-малко число - условие

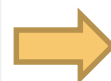
- Напишете програма, която:
 - Получава цяло число(*n*) от потребителя
 - Взима числа *n* последователни пъти
 - Намира най-малкото измежду тях
- Примерен вход и изход:

2
99
100



99

3
-10
20
-30



-30

4
45
-20
7
99



-20



Най-малко число - решение

```
let n = Number(input[0]);  
let counter = 0;  
let min = Number.MAX_SAFE_INTEGER;  
while (counter < n) {  
    //TODO: Use logic similar to the previous problem  
}
```

- Напишете програма, която:
 - Изчислява **средната оценка** на ученик от цялото му обучение
 - Ако годишната му оценка е:
 - **≥ 4.00** , ученикът преминава в следващия клас
 - **< 4.00** , той ще повтори класа
 - При **завършване** да се отпечата:
"{име на ученика} graduated. Average grade: {средната оценка от цялото обучение}"

Завършване - условие (2)

- Примерен вход и изход:

Pesho
4
5.5
6
5.43
4.5
6
5.55
5
6
6
5.43
5



Pesho graduated.
Average grade: 5.37

Ani
5
5.32
6
5.43
5
6
5.5
4.55
5
6
5.56
6



Ani graduated.
Average grade: 5.45

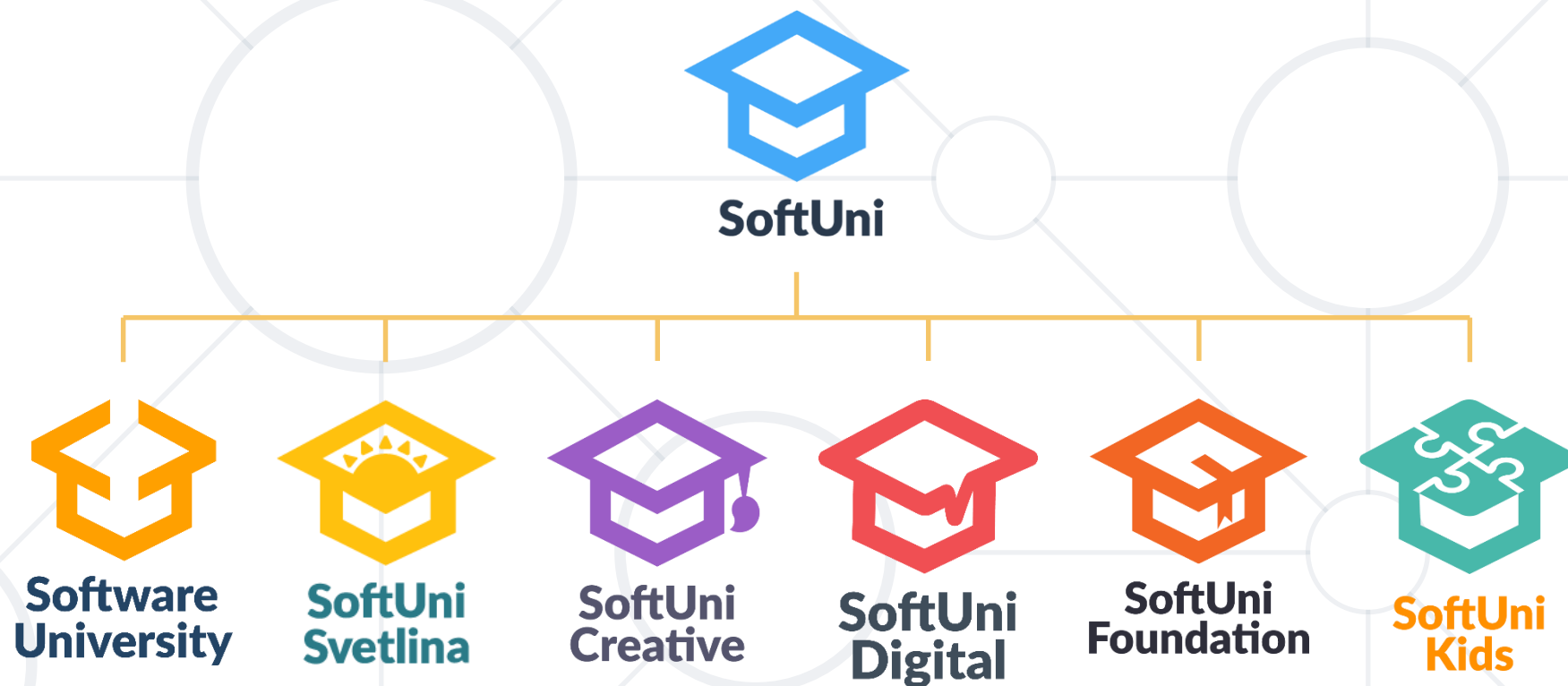
```
function graduation(input) {  
  let name = input.shift();  
  let counter = 1;  
  let sum = 0;  
  while (counter <= 12) {  
    let grade = Number(input.shift());  
    if (grade >= 4.00) {  
      sum += grade;  
      counter++;  
    }  
  }  
  let average = sum / 12;  
  //TODO: print the output  
}
```



- Можем да инкрементираме/декрементираме числови стойности
- Използваме **while** - цикли, за да повтаряме действие, докато е в сила дадено условие
- Можем да прекъсваме циклите с оператора **break**



Въпроси?



SoftUni Diamond Partners



XSsoftware



SBTech
we know sports



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твоето утре



INDEAVR

Serving the high achievers



INFRAGISTICS®



STEMO®
Computer Systems & Software

SUPERHOSTING.BG

SoftUni Organizational Partners

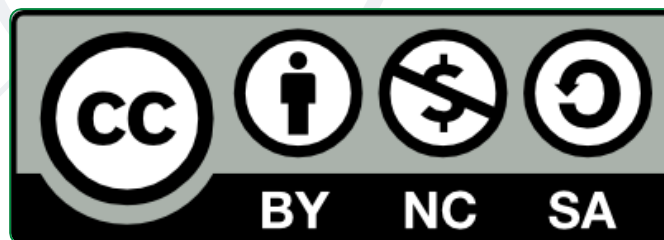


OneBit
SOFTWARE



WORLD
OF
MYTHS

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"



- Благодарности: настоящият материал може да съдържа части от следните източници
 - Книга "Основи на програмирането с JavaScript" от Светлин Наков и колектив с лиценз CC-BY-SA

Обучения в СофтУни

- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
- Software University Foundation
 - <http://softuni.foundation/>
- Software University @ Facebook
 - [facebook.com/SoftwareUniversity](https://www.facebook.com/SoftwareUniversity)
- Software University Forums
 - forum.softuni.bg



**Software
University**

