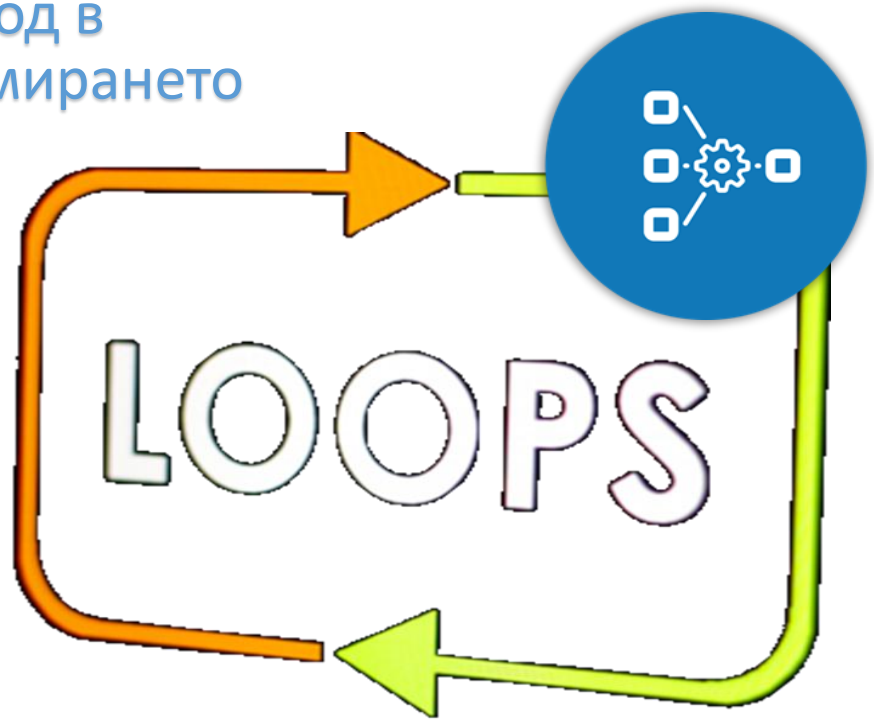
Други видове цикли

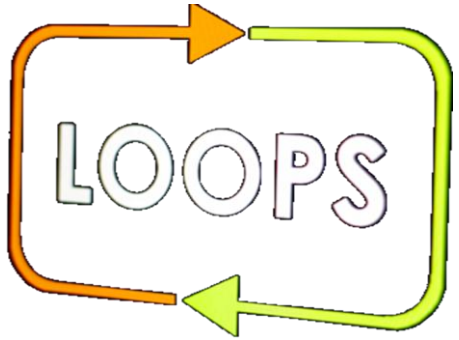
while u do-while

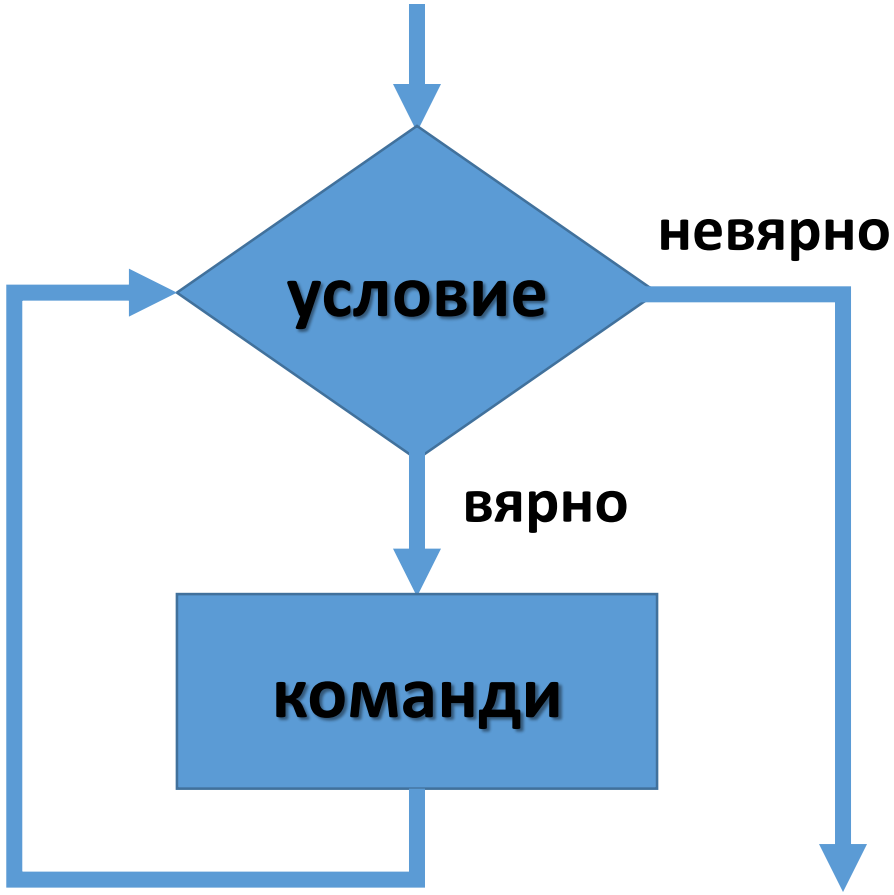
Увод в
програмирането



Съдържание

- Други конструкции за цикъл:
  - `while` цикъл
  - `do-while` цикъл
- Безкраен цикъл
- Оператор `break` и `continue`
- По-сложни задачи с вложени цикли





Конструкция `while`

Повторение, докато е в сила дадено условие

Редица числа 2k+1

- Да се отпечатат всички числа ≤ **n** от редицата: 1, 3, 7, 15, 31, …
  - Всяко следващо число = предишно число * 2 + 1

```
var num = 1;
while (num <= n)
{
    Console.WriteLine(num);
    num = 2 * num + 1;
}
```

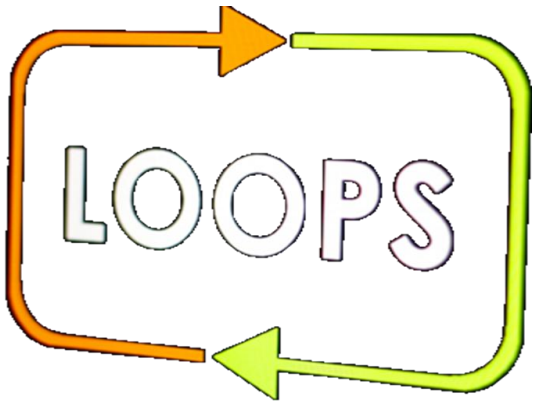
Повтаряй докато е в сила условието num ≤ n

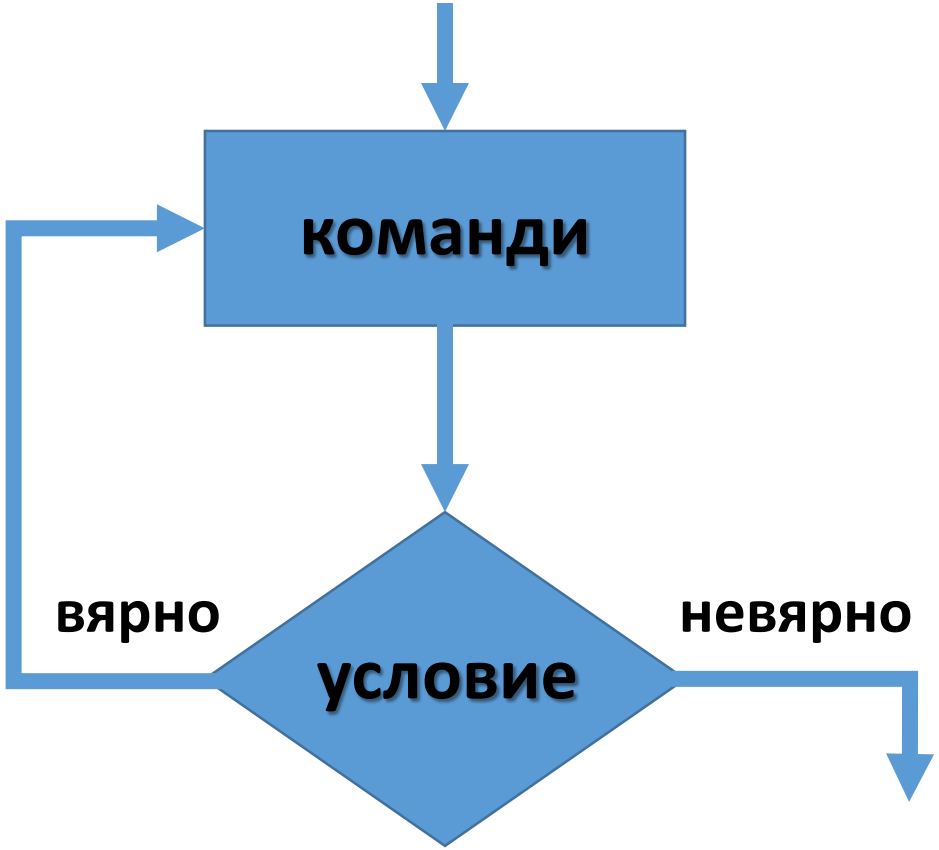
1, 3, 7, 15, 31, 63, …

Число в диапазона [1…100]

- Да се въведе число в диапазона [1…100]
  - При невалидно число да се въведе отново

```
var num = int.Parse(Console.ReadLine());
while (num < 1 || num > 100)
{
    Console.WriteLine("Invalid number!");
    num = int.Parse(Console.ReadLine());
}
Console.WriteLine("The number is: {0}", num);
```





Do...While цикъл

Повторение, докато е изпълнено условието

Изчисляване на факториел

- За естествено число **n** да се изчисли **n!** = **1** * **2** * **3** * … * **n**
  - Пример: **5!** = **1** * **2** * **3** * **4** * **5** = **120**

```
var n = int.Parse(Console.ReadLine());
var fact = 1;
do
{
    fact = fact * n;
    n--;
} while (n > 1);
Console.WriteLine(fact);
```

Сумиране на цифрите на число

- Да се сумират цифрите на цяло положително число **n**
  - При **n** = 5634: 5 + 6 + 3 + 4 = 18

```
var n = int.Parse(Console.ReadLine());
var sum = 0;
do
{
    sum = sum + (n % 10);
    n = n / 10;
} while (n > 0);
Console.WriteLine("Sum of digits: {0}", sum);
```

n % 10 връща последната цифра на числото **n**

n / 10 изтрива последната цифра на **n**

Безкрайни цикли и оператор break

Безкраен цикъл

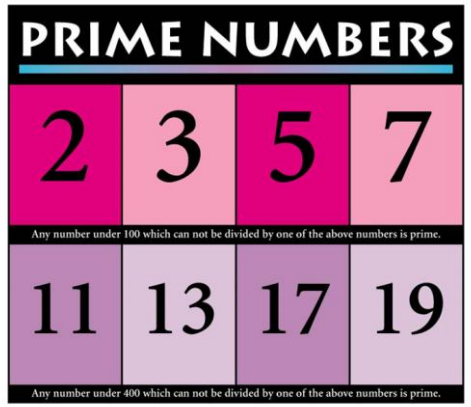
- Безкраен цикъл е когато повтаряме нещо до безкрайност:

```
while(true)
{
    Console.WriteLine("Infinite loop");
}
```

```
for (;;)
{
    Console.WriteLine("Infinite loop");
}
```

Прости числа

- Едно число **n** е просто, ако се дели единствено на **1** и **n**
  - Прости числа: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, …
  - Непрости (композитни) числа: 10 = 2 * 5, 21 = 3 * 7, 143 = 13 * 11
- Едно число **n** е просто, ако се дели на число между **2** и **n-1**
- Алгоритъм за проверка дали число е просто:
  - Проверяваме дали **n** се дели на 2, 3, …, n-1
  - Ако се раздели, значи е композитно
  - Ако не се раздели, значи е просто



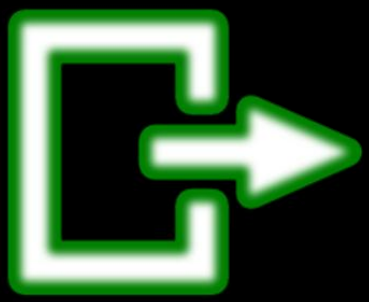
- Оптимизация: вместо до n-1 да се проверяват делители до $\sqrt{n}$

Проверка за просто число. break и continue

```
var n = int.Parse(Console.ReadLine());
var prime = true;
for (var i = 2; i <= Math.Sqrt(n); i++)
    if (n % i == 0) {
        prime = false;
        break;
    }
if (prime) Console.WriteLine("Prime");
else Console.WriteLine("Not prime");
```

има и команда **continue** - за следващо повторение

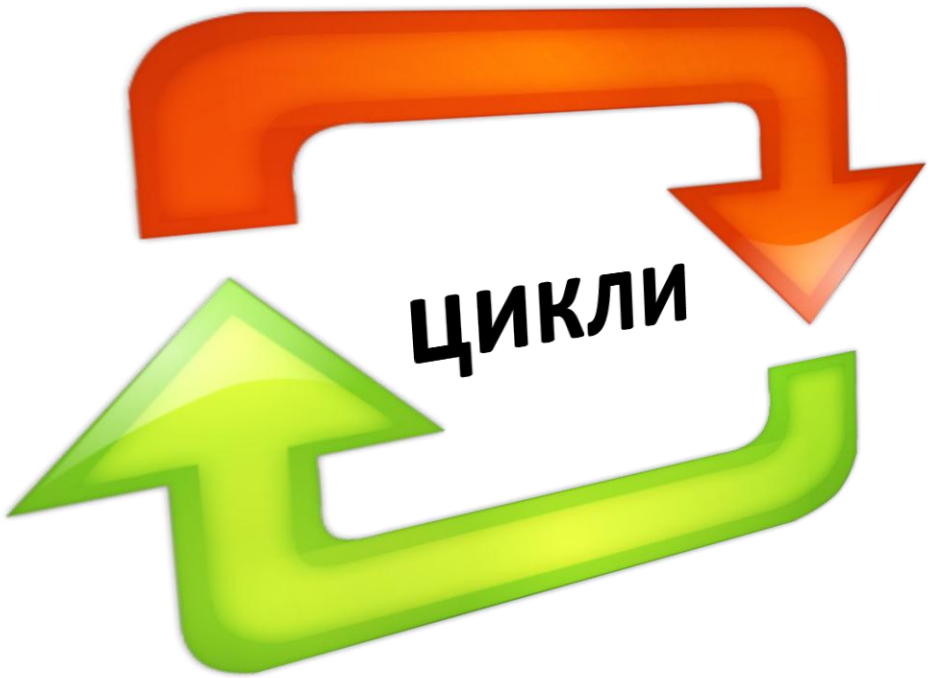
break излиза от цикъла



Оператор break в безкраен цикъл

- Да се напише програма, която въвежда четно число
  - При невалидно число да връща към повторно въвеждане

```
var n = 0;
while (true) {
    Console.Write("Enter even number: ");
    n = int.Parse(Console.ReadLine());
    if (n % 2 == 0)
        break; // четно число-> изход от цикъла
    Console.WriteLine("The number is not even.");
}
Console.WriteLine("Even number entered: {0}", n);
```



Задачи с цикли

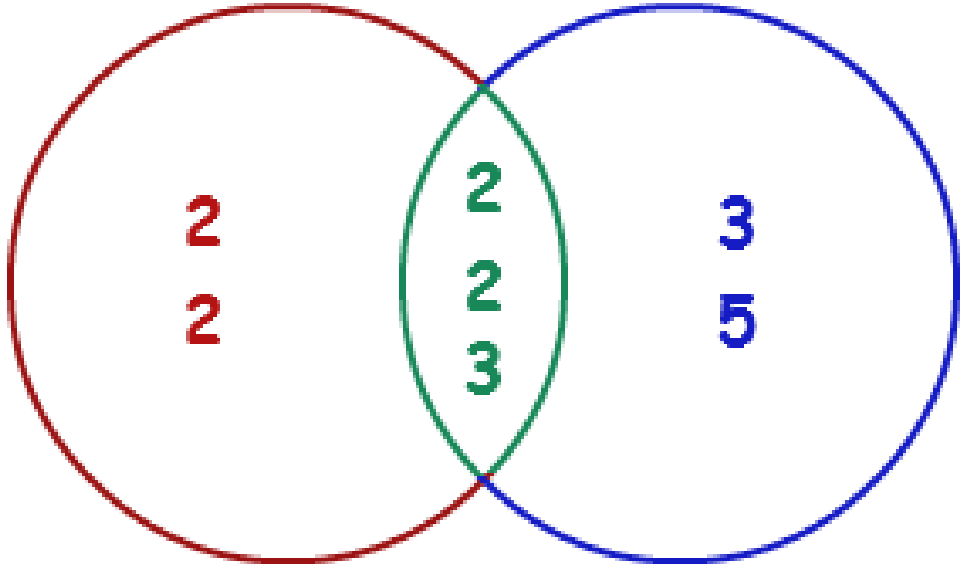
Числа на Фибоначи

- Числата на Фибоначи са следните: 1, 1, 2, 3, 5, 8, 13, 21, 34, …
  - $F_0 = 1$
  - $F_1 = 1$
  - $F_n = F_{n-1} + F_{n-2}$

Пример: F(15) = 987

Да се въведе n и да се пресметна n-тото число на Фибоначи

```
var n = int.Parse(Console.ReadLine());
var f0 = 1;
var f1 = 1;
for (var i = 0; i < n-1; i++)
{
    var fNext = f0 + f1;
    f0 = f1;
    f1 = fNext;
}
Console.WriteLine(f1);
```



Алгоритъм на Евклид

Най-голям общ делител (НОД)

Най-голям общ делител (НОД)

- Най-голям общ делител (НОД) на две естествени числа **a** и **b** е най-голямото число, което дели едновременно **a** и **b** без остатък
  - НОД(24, 16) = 8
  - НОД(67, 18) = 1
  - НОД(12, 24) = 12
  - НОД(15, 9) = 3
  - НОД(10, 10) = 10
  - НОД(100, 88) = 4
- Алгоритъм на Евклид за намиране на НОД:
  - Докато не достигнем остатък `0`
    - Делим по-голямото число на по-малкото
    - Взимаме остатъка от делението

```
while b ≠ 0
  var oldB = b;
  b = a % b;
  a = oldB;
print a;
```

Алгоритъм на Евклид за НОД

- Да се въведат цели числа **a** и **b** и да се намери **НОД**(**a**, **b**)

```
var a = int.Parse(Console.ReadLine());
var b = int.Parse(Console.ReadLine());
while (b != 0)
{
    var oldB = b;
    b = a % b;
    a = oldB;
}
Console.WriteLine("GCD = {0}", a);
```

Какво научихме днес?

- **while** повтаря докато е вярно условието:

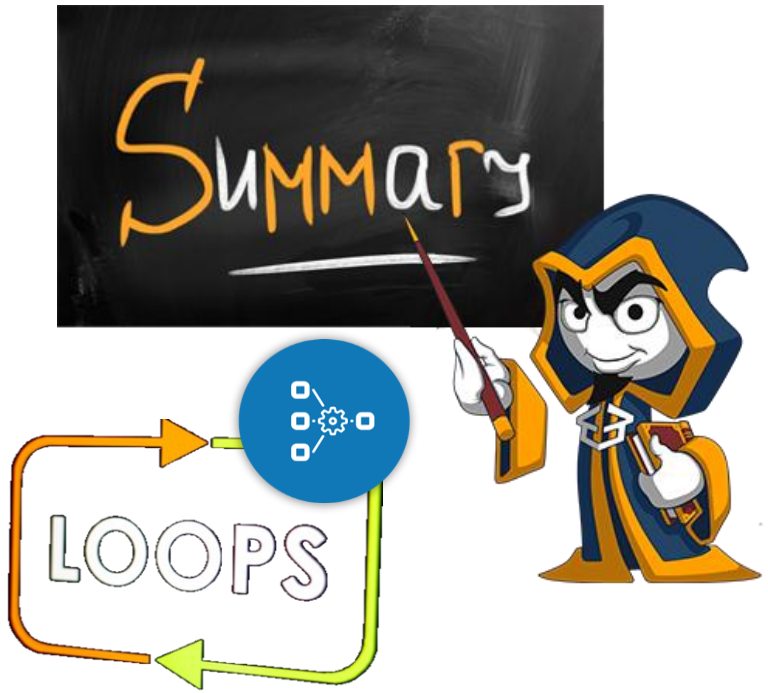
```
int num = 1;
while (num <= n)
    Console.WriteLine(num++);
```

- Безкраен цикъл:

```
for (;;)
{ Console.WriteLine("Infinite loop"); }
```

- Прекъсване на безкраен цикъл – с **break**

```
while(true) { break;
  Console.WriteLine("This will not execute");
}
```



Други видове цикли

Въпроси?

Договор за ползване

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

Базиран е на учебните материали на НП „Обучение за ИТ Кариера“.

Може да съдържа части от следните източници:

- Книга "Основи на програмирането със C#" от Светлин Наков и колектив с лиценз CC-BY-SA