# C#

## Zanke – while zanka

# Izpiši naravna števila od 1 do 15

- Kako do izpisa
  - Rešitev: tipkanje števil

```
string odg = "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15";
Console.WriteLine(odg);
```

- Rešitev je sicer pravilna! Pa vendar: kaj pa, če bi želeli izpisati števila med 1 in 1000?

# Še en primer: kaznovani Janezek

- Janezek je bil v šoli poreden. Zato mu je učiteljica naročila, naj sestavi program, ki 5x izpiše "*V šoli se moram lepo obnašati!*“
- Janezek hitro vzame "okostje" programa in ga dopolni s
  - *Console.WriteLine("V šoli se moram lepo obnašati!");*
  - *Console.WriteLine("V šoli se moram lepo obnašati!");*
  - *Console.WriteLine("V šoli se moram lepo obnašati!");*
  - *Console.WriteLine("V šoli se moram lepo obnašati!");*
  - *Console.WriteLine("V šoli se moram lepo obnašati!");*
- Ker pa se Janezek še vedno ne obnaša lepo, sledi nova kazen. Ker učiteljica ve, da je podravnatelj, ki na šoli med drugim opravlja tudi nalogo vzdrževalca programske opreme, uspel "sesuti" operacijski sistem tako, da se enostavno ne da več kopirati besedila, Janezku pa ni nič bolj zoprnega kot veliko tipkanja, je kazen vzgojna. Sedaj mora 100x izpisati omenjeni stavek.
- Janezek je sprva obupan. Ampak bistra glavica, v Google natipka *"*C# ponavljanje istih ukazov" in kaj hitro se mu razvedri čelo …

# Zanke

- Seštej 10 števil, izpiši 20 zvezdic, nariši n krogov, seštej dosežene točke vsakega dijaka, izračunaj plačo zaposlenih, …

- Kaj je skupno vsem tem problemom. Gre za **ponavljanje**. Izvajamo isti postopek le s spremenjenimi podatki. Ponavljanju stavkov v programiranju pogosto rečemo **zanke**.
  - isti postopek, spremenjeni podatki
  - Vrste zank: `while, for, do while` in `foreach`

- osnovna zanka
  - `while`

# while

- Sintaksa:

```
while (pogoj)
{
    stavek1;
    ...
    Stavek_n;
}
```

Stavki, ki se ponavljajo

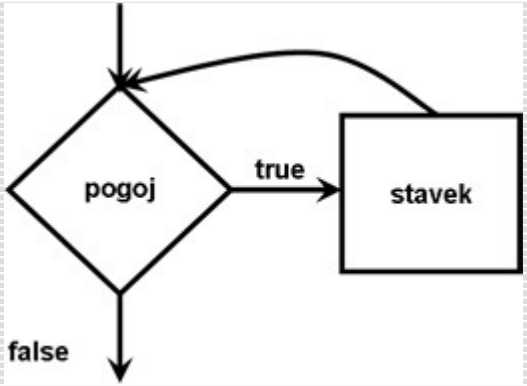
- Izvajanje
  - Preveri pogoj. Če je resničen, izvedi stavke v zavitih oklepajih.
  - Preveri pogoj. Če je resničen, ponovno izvedi stavke.
  - Preveri pogoj …

- Dokler je logični pogoj izpolnjen, se izvajajo stavki med zavitima oklepajema.

# While – shematski prikaz

- Ponavljaj:

```
če je pogoj true,
        izvedi stavek,
sicer prekini izvajanje
```



# While – značilnosti zanke

- Najpomembnejše značilnosti zanke **while** so
  - pogoj preverimo na začetku zanke,
  - zanka se izvaja, dokler je pogoj izpolnjen,
  - če pogoj ni izpolnjen že na začetku, se zanka ne izvede niti enkrat,
  - pogoj, ki ga preverjamo, je lahko sestavljen.

- Na dolgo bi delovanje *while* zanke opisali takole:
  - najprej se preveri pogoj,
  - če ima pogoj vrednost *true* (je resničen), se izvede stavek,
  - nato se ponovno preveri pogoj,
  - če je še vedno izpolnjen, se znova izvede stavek,
  - ponovno se preveri pogoj,
  - ...
  - če pogoj ob preverjanju nima več vrednosti *true* (ni več resničen), se konča izvajanje zanke *while*.

# Zgled - Janezek

- kaj ponavljamo:
  - izpis stavka
- pogoj:
  - dokler ne izpišemo 100 stavkov
  - torej moramo šteti izpisane stavke
  - Pogoj: (izpisanihStavkov < 100)
- pred začetkom ponavljanja določimo števec ponavljanj
  - izpisanih stavkov še ni
    - izpisanihStavkov = 0;
- v zanki
  - izpišemo stavek
  - števec izpisanih stavkov povečamo za 1

```
izpisanihStavkov = izpisanihStavkov + 1;
```

# Janezek : program (del)

```
int izpisanihStavkov = 0; //začetna vrednost števca

while (izpisanihStavkov < 100)
{
   Console.WriteLine("V šoli se moram lepo obnašati!");
   // nov izpisani stavek
   izpisanihStavkov = izpisanihStavkov + 1;
}
```

# Kvaliteta računalniške kocke

- Zanima nas, koliko dobra je "računalniška kocka". V ta namen bomo računalniku naročili naj 6 000 000 krat vrže kocko in šteje, koliko je vrgel denimo 2 ali pa 6 pik. Če je kocka "poštena", se obe vrednosti ne smeta veliko razlikovati od 1 000 000.
- Zanka – uporabimo npr. zanko while
  - znotraj zanke izvedemo en met
  - če je met 6 ali 2, povečamo ustrezni spremenljivki

# Kvaliteta računalniške kocke : program

```
int stevecMetov = 1;
int stevec2 = 0; // kolikorat smo vrgli 2
int stevec6 = 0; // kolikorat smo vrgli 6

Random nakl = new Random();//generator naključnih števil

while (stevecMetov <= 6000000)
{
   int kocka = (nakl.Next(1,7)); //računalniški met kocke
   if (kocka == 2) //preverimo, če je izid enak 2
   {
     stevec2 = stevec2 + 1; // stevec2++;
   }
   if (kocka == 6) //preverimo, če je izid enak 6
   {
    stevec6 = stevec6 + 1;
   }
   stevecMetov = stevecMetov + 1; // nov met
}
```

# While – zgled

```
while (x>1)
{
   x = x / 2;
}
```

- Kaj se zgodi, če je v x na začetku vrednost 4.2
- ustavitev
- zaciklanje

# While – zgled

```
while (x <= 10)
{
   x = x + 1;
}
```

- Kaj se dogaja, če je v x na začetku 1?
- Kaj se dogaja, če je x na začetku 100?

# Izpiši števila od 1 do 20

- Števec od 1 do 20
- Zanka
  - Pogoj: dokler je števec manjši ali enak 20

    ```
    while (stevec <= 20)
    ```

- Na vsakem koraku
  - števec dodamo k izpisu

    ```
    izpis = izpis + " " + stevec;
    ```
  - Povečamo števec

    ```
    stevec = stevec + 1;
    ```

# Izpiši števila od 1 do 20 - program

```
string izpis = "";
int stevec = 1;

while (stevec <= 20)
{
    izpis = izpis + " " + stevec;
    stevec = stevec + 1;
}
Console.WriteLine(izpis);
```

# Števec ponovitev

- Napisati želimo zanko, v kateri se bo telo zanke izvedlo tolikokrat, kot želimo

```
stevec = 1; //začetna vrednost števca
while (stevec <= stevilo_ponovitev)
{
    ... //poljubno število stavkov: naredimo nekaj
    stevec = stevec + 1;
  //povečamo števec
}
```

  - Števec je zaporedoma: `1, 2, 3, ..., stevilo_ponovitev`
  - Zanka je zaključena, ko števec preseže vrednost spremenljivke `stevilo_ponovitev`

# Izpis števil od a do b

- Izpiši števila od a do b (a in b podatka, ki ju preberemo)
  - Ne moremo narediti z neposrednim izpisom
  - obvezna zanka

- Potrebujemo:
  - Števec od a do b
  - Začetna vrednost števca: a

    ```
    stevec = a;
    ```
  - Zanka
    - Pogoj: dokler je števec manjši ali enak b

    ```
    while (stevec <= b)
    ```
  - Na vsakem koraku
    - števec dodamo k izpisu

      ```
      izpis = izpis + " " + stevec;
      ```
    - Povečamo števec

      ```
      stevec = stevec + 1;
      ```

# Izpis števil od a do b - program

```
string izpis = "";
int od_kje,
    do_kam; // "do" ne bi bilo dobro ime - rezervirana beseda!
int stevec;  //števec ponavljanj
string beri;
//branje podatkov
Console.Write("Od kje naprej: ");
beri = Console.ReadLine();
od_kje = Convert.ToInt32(beri);
Console.Write("Do kam: ");
beri = Console.ReadLine();
do_kam = Convert.ToInt32(beri);

//zanka
stevec = od_kje;
while (stevec <= do_kam)
{
   izpis = izpis + " " + stevec; // dodajamo k izpisu
   stevec = stevec + 1;  //povečujemo števec ponavljanj
}
Console.WriteLine(izpis);
```

# Koliko metov do šestice

- Sestavimo program, ki bo ugotovil, koliko metov smo morali vreči, da smo vrgli 6.
- Potrebujemo
  - Generator naključnih števil

    ```
    Random = new Random();
    ```
  - pogoj: vrženi met je različen od 6

    ```
    met != 6
    ```
  - kaj ponavljamo
    - met kocke

      ```
      met = (nakl.Next(1,7));
      ```
    - štetje metov

      ```
      stevecMetov = stevecMetov + 1
      ```
  - Kako začeti
    - dogajanje pred zanko: inicializacija števca metov

      ```
      stevecMetov = 0;
      ```
    - kako "vstopiti" v zanko
      - pogoj mora biti izpolnjen `(met != 6)`
      - zato postavimo spremenljivko `met` na karkoli (razen na 6 seveda)!

# Koliko metov do šestice : program

```
int stevecMetov = 0;
Random nakl = new Random(); // generator naključnih števil
int met = 0; // karkoli, le 6 ne
// mečemo, dokler ne vržemo 6
 while (met != 6)
{
     met = (nakl.Next(1,7)); // nov met kocke
     stevecMetov = stevecMetov + 1;
 }
// konec zanke, ker smo vrgli 6
string odg = "Do šestice je bilo potrebno " + stevecMetov;

odg = odg + " metov!";
Console.WriteLine(odg);
```

# Kaj počne tale del programa

```
int i = 100;
string odgovor = "";
while (i > 100)
{
    odgovor = odgovor + "riba raca rak ";
    i = i + 1;
}
Console.WriteLine(odgovor);
```

# Kaj pa tale

```
string odgovor = "";
while (true)
{
  odgovor = odgovor + "klop pod klopjo, ";
  Console.WriteLine(odgovor);
}
 Console.WriteLine("Končali smo...");
```

# In ta?

```
string odgovor = "";
i = 1;
while (i <= 10)
    odgovor = odgovor + "klop pod klopjo, ";
    Console.WriteLine(odgovor);
    i = i + 1;
Console.WriteLine("Končali smo ...");
```

# In ta?

```
string odgovor = "";
int i = 1;
while (i <= 10);
{
     odgovor = odgovor + "klop pod klopjo, ";
     Console.WriteLine(odgovor);
     i = i + 1;
}
Console.WriteLine("Končali smo ...");
```

# Morebitne težave

- Pogoj napačen
  - Zanka se nikoli ne konča!
- Pozabili smo na oklepaja { }
  - Oklepaja pišemo tako kot pri pogojnem stavku
  - V telesu le en stavek – oklepaja nista potrebna
  - Običajno pa želimo narediti več stvari (sestavljeni stavek) – pišemo oklepaja
  - POZOR! Zamikanje ne pomaga (prevajalniku je vseeno)
- Pogosta napaka: zapis ; (podpičje!) takoj za pogojem
  - Zanka, ki ima v telesu "prazen" stavek

```
while (pogoj); //primer prazne zanke while
```

# Kako sestavljamo program z zanko

- Premislimo, kaj se dogaja v splošnem
  - Tekoča ponovitev zanke
    - Pišemo i-ti krog
    - Pregledujemo tekočo vrstico
    - ...
- Dogajanje na začetku (pred vstopom v zanko)
  - Posebni pogoji ...
  - Nastavitev števcev
  - vrednost kontrolne spremenljivke taka, da se zanka sploh začne, ...
- Dogajanje na koncu
  - Ali je potrebno z zadnjim elementom kaj posebnega narediti
  - Smo števec po "nepotrebnem" preveč povečali
  - ...

# Zanka while – ponavlja se en sam stavek

- ponavljanje stavka (stavkov) v odvisnosti od pogoja

```
while (<pogoj>)
   stavek
```

- POZOR: obvezna oklepaja () okoli pogoja

- preverimo `pogoj`. Če je `pogoj` izpolnjen, izvedemo `stavek`. Ponovno preverimo pogoj, …

# Zanka while – ponavlja se več stavkov

- ponavljaj `stavke`, dokler je `pogoj` izpolnjen (ima vrednost `true`)

```
while (<pogoj>)
{
    stavek1;
    …
    stavekn;
}
```

```
while (stevilkaPoskusa < n)
{
  izvediPoskus(stevilkaPoskusa);
  stevilkaPoskusa++; // stevilkaPoskusa = stevilkaPoskusa + 1;
}
```

# Zgledi

Igra “buuum”

Ugibanje števil

V Butalah menjajo valuto

# Igra “Buuum”

- "Buuum" je igra z naslednjim pravilom: Sodelujoči zaporedoma govorijo števila, vendar morajo namesto večkratnikov števila pet in večkratnikov števila sedem reči *buuum*. Sestavimo program, ki bo po tem pravilu izpisal cela števila od a do b.

- Pomagali si bomo s pogojnim stavkom, ki preveri, ali je število večkratnik števila 5 ali števila 7. Ko bo pogoj pogojnega stavka resničen, bomo izpisali besedo *buuum*.

# Igra “Buuum”

```
Console.Write("Vnesite a: "); // spodnja mehja intervala
int a = Convert.ToInt32(Console.ReadLine());
Console.Write("Vnesite b: ");
int b = Convert.ToInt32 (Console.ReadLine());
// Drugi vnos mora biti manjši od prvega
if (b < a)
{
      int t = a;
      a = b;
      b = t;
}
// Izpis besede buuum ali števil
while(a <= b)
{
   if((a % 5 == 0) || (a % 7 == 0))
      Console.Write("buuum"); //en sam stavek oklepaja {} NISTA potrebna

      else
         Console.Write(a);
      Console.Write(" ");
      a = a + 1;
}
```

# Ugibanje števila

- Računalnik naj si "izmisli" število med 1 in 100! Koliko korakov bomo potrebovali, da ga uganemo? Računalnik nam bo odgovarjal z "hladno", če bomo za več kot 15 proč, s "toplo", če se bo naše število od računalnikovega razlikovalo za 14 – 6 in "vroče", če bomo le še največ 5 oddaljeni od števila. Seveda bo povedal tudi, če bomo število uganili!
- Ali vemo koliko bo ponovitev?
  - ne, število ponovitev ne kontrolira števec, ampak naša uspešnost ugibanja!

# Ugibanje števila - zanka

- Logična spremenljivka: `aliSmoUganili = false;`
- Zanka:
  - `while (aliSmoUganili == false)`
  - ali `while (!aliSmoUganili)`
- V zanki:
  - vnos našega števila
  - Preverimo razliko

    ```
    razlika = Math.Abs(naseStevilo – racStevilo);
    if (razlika == 0) // uganili smo!
    ```

    - Če je razlika enaka 0 izpišemo obvestilo in končamo zanko: `aliSmoUganili = true;`
  - `if (razlika <= 5)`
    - Izpišemo "Vroče"
  - `if ((razlika > 5) && (razlika < 15))`
    - Izpišemo "Toplo"
  - `if (razlika >= 15)`
    - Izpišemo "Hladno"

# V Butalah menjajo valuto

- V Butalah je prava panika. 1. januarja bodo butalske cekine zamenjali za novo valuto, še vedno pa ni znano, kakšen bo menjalni tečaj. No, panika je odveč, saj so iz Banke Butale sporočili, da so vladni modreci odločili, da bo menjalni tečaj odvisen od obnašanja Šprince Marogaste, glavne butalske uši, opoldne na Slivestrovo. Tako bodo imeli celo popoldne in celo noč, da bodo lahko pripravili nove cenike.
- No, na srečo pa so v Butalah v veljavi samo cene 50, 100, 150, 200, …, 1000 cekinov. Zato bo glavni butalski informatik pripravil program, ki bo takoj, ko bo znan menjalni tečaj, za vse te cene izpisal njihovo vrednost v novi valuti.
- Ampak razbojnik Cefizelj je ravno tiste dni ukradel edino miško v Butalah. Zato pomagaj Butalcem in ti sestavi ustrezni program. Pri tem upoštevaj, da bo menjalni tečaj dan na decimalke, a nova valuta pozna le cele vrednosti. A pozor, Butalci imajo malo drugačno zaokrožanje – če so desetinke sode, se zaokroži navzdol, če pa lihe, pa navzgor.

# V Butalah menjajo valuto : ideja

- za vsako ceno je potrebno narediti isti postopek
  - izračunati vrednost v novi valuti
  - pravilno zaokrožiti
  - Izpisati vrednost v cekinih in novi valuti
- cene se lepo "urejeno" spreminjajo
  - naraščajo po 50
  - while (cena <= 1000)
    - cena = cena + 50;

# V Butalah menjajo valuto : program

```
int cena = 50; // najnižja cena
double novaVrednost; // pretvorjena cena v novi valuti
int pravaNovaVrednost; // cena v novi valuti
... Beremo ...
double menjalniTecaj = double.Parse (...
while (cena <= 1000) { // cene so manjše od 1000
   novaVrednost = cena / menjalniTecaj;
   // zaokrožanje
   int zacVred = (int)(novaVrednost * 10);
   int enice = zacVred % 10;
   pravaNovaVrednost = zacVred / 10;
   if (enice % 2 == 1) { // lihe desetinke, popravek navzgor
     pravaNovaVrednost = pravaNovaVrednost + 1;
   }
   Console.WriteLine …
   cena = cena + 50; // nova cena
}
```