C#- Loopar

Vad är en loop och vad används de till?

Kortfattat så används en loop när vi vill köra en kodsats flera gånger eller när vi exempelvis vill iterera igenom en array. Vi kommer nu gå igenom fyra loopar, For-loopen, While-loopen, Do…While-loopen och Foreach-loopen.

For-loopen

For-loopen används när man behöver gå igenom kod ett bestämt antal gånger. Man deklarerar en for-loop genom att först skriva nyckelordet ”for” följt av parenteser. Inom parentesen skriver vi vårt argument. Först deklarerar och tilldelar vi en variabel som används som en räknare, därefter ett villkor och sedan avslutar vi argumentet med att öka vår räknare med 1.

Exempel:

```
for(int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

I koden ovan skapar vi en for-loop som ska itereras 10 gånger. Först deklarerar vi variabeln *i* och tilldelar den värdet 0. Därefter sätter vi villkoret att så länge som *i* är lägre än 10 så ska koden inuti köras. Därefter ökar vi vår räknare *i* med 1 efter varje avslutad loop. Loopen i exemplet skriver ut vår räknares värde och går från 0 till 9, det vill säga att värdet skrivs ut 10 gånger. Det är en standard att döpa räknar-variabler till bokstaven ”i” och fortsätter längs alfabetet om man skulle nästla en loop, dock kan man döpa dessa variabler till vad man vill.

Nästlade for-loopar

Det är inte helt ovanligt att man behöver nästla sina loopar. För att nästla en loop så deklarerar man ännu en loop inuti sin nuvarande loop. Detta betyder att för varje iteration av första loopen så kommer den andra loopen iterera tills den är klar.

Exempel:

```
for (int i = 1; i < 6; i++)                    // Loop 1
{
    for (int j = 0; j < i; j++)                // Loop 2
    {
        Console.Write(i);
    }
    Console.WriteLine(); // Ny rad
}
```

Exemplet ovan ser ni en nästlad for-loop. Loop 1’s uppgift är att iterera Loop 2 fem gånger och för varje loop skriver den även ut en ny rad eftersom Console.Write i loop 2 skriver ut på samma rad. Loop 2 kör sin kod så länge som *j* är lägre än *i.* Första iterationen så kommer *i* ha värdet 1, och *j* kommer ha värdet 0. Det vill säga att loop 2 kommer att köras en gång och skriva ut en etta sedan avslutas loop 2, loop 1 skriver ut en ny rad och ökar räknaren. Andra iterationen så kommer *i* ha värdet 2 vilket betyder att loop 2 kommer att iterera två gånger,

så den skriver ut en tvåa två gånger, sedan hoppar den ur loopen och skriver ut en ny rad. När vi har itererat klart så kommer output att se ut så här:

Exempel:
```
1
22
333
4444
55555
```

While-loopen

En While-loop används när man vill iterera igenom en kod så länge ett villkor är sant. Vi säger att vi ska skapa en meny till ett program och vi vill att programmet ska köras så länge som en användare inte har matat in siffran 0, då kan det se ut så här:

Exempel:
```
int userChoice = -1;
while (userChoice != 0)
{
    Console.WriteLine(”Fiktiv meny med 4 val”);
    Console.Write(”Ditt menyval, 1-3 eller 0 för att avsluta: ”);
    userChoice = Console.ReadLine();
}
```

Här skapar vi först en variabel som vi använder när vi skapar vårt villkor. Därefter skapar vi våran while-loop och sätter villkoret userChoice != 0 som man kan förklara genom att säga ”så länge som userChoice inte är lika med noll”. I loopen skriver vi ut vår ”meny” samt ber användaren att mata in ett val och valet sparas i userChoice. Därefter kollar loopen sitt villkor igen, och så länge som userChoice inte är 0 så kör den kodsatsen igen.

Do…While-loopen

Do…while-loopen fungerar precis som en while-loop fast med ett undantag, den kör alltid koden minst en gång. Om vi använder samma exempel som tidigare med en meny så ser det ut så här:

Exempel:
```
int userChoice = -1;
do
{
    Console.WriteLine(”Fiktiv meny med 4 val”);
    Console.Write(”Ditt menyval, 1-3 eller 0 för att avsluta: ”);
    userChoice = Console.ReadLine();
}
while(userChoice != 0);
```

En do..while-loop kan man förklara genom att säga ”Kör den här koden… så länge villkoret stämmer”.

Foreach-loopen

Vi kan använda oss av en foreach-loop när vi vill iterera igenom alla element i en *array* eller samling. En foreach-loop använder sig inte utav någon form av indexhantering utan den loopar igenom en samling efter turordning. Därför är en foreach-loop den föredragna loopen att använda för *arrayer* och andra samlingar då vi eliminerar fel som kan uppstå när vi använder oss av index, men i sin tur ger oss mindre flexibilitet. Vi använder oss ett exempel där vi har en sträng-array med 3 element:

Exempel:

```
string[] dogs = new string[3] {”Fido”, ”Charlie”, ”Stefan”};
foreach (string hundNamn in dogs)
{
    Console.WriteLine(hundNamn);
}
```

För att beskriva en foreach med ord kan man uttrycka sig med orden ”för varje(datatyp variabelNamn i samling). I detta fall blir datatypen en string, namnet blir hundNamn och samlingen blir dogs. Just namnet i en foreach-loop kan ses som ett objekt (mer om det i andra kursen) och kan ha sina medföljande metoder. Låt oss tänka att samling objekt vi kallar ”Personer” med en metod som vi kallar ”Beskrivning” som i sin tur skriver ut en kort beskrivning om varje person, då hade vi kunnat anropa metoden på alla ”Personer” i samlingen.

Exempel:

```
foreach (Personer person in PersonSamling)
{
    person.Beskrivning();
}
```

Denna loop kan man förklara genom att säga ”För varje person i PersonSamling, anropa metoden Beskrivning()”.