Programmeren in C#
les 2: programmaverloop

Samenvatting lesavond 1
Lineair programmaverloop

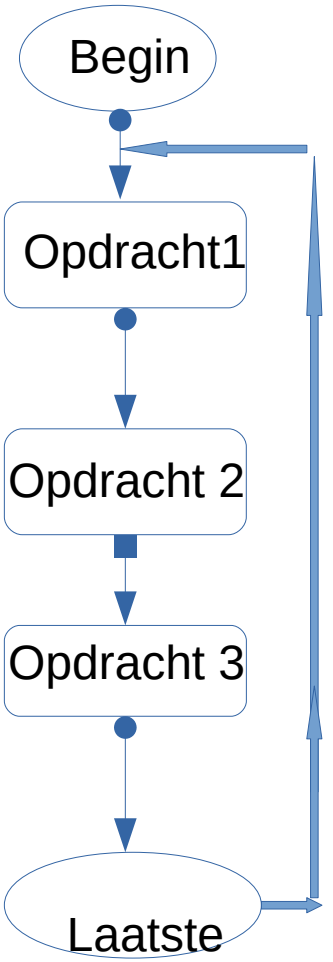
Deel 1 anatomie v.e. programma

1) structuur
2) syntax
3) variabelen
4) lineair programmaverloop
5) instructies en operatoren

Deel 2 functies

1) tijdsfuncties
2) monitor functies
3) I/O functies
4) random functies

5) home-brewd



Structuur:

SETUP() en LOOP()

Syntax:

scheidingstekens **; en {}**

Variabelen:

naamgeving
type
declaratie

Operatoren:

toekenning **=, >, <**
rekenkundige **+, –, *, /**
logische: bitgewijze **&, |, ~ ,<<, >>**
booleaanse; **&&, ||, !**,
vergelijkende: **==, >, <**

Tijdfuncties:

Delay() ; mills(); micro()

Monitor functies:

Serial.begin()
Serial.print()
Serial.println()

I/O-functies:

pinMode()

digitalRead() ; DigitalWrite()
analogRead() ; AnalogWrite()

Random functies:

random(); randomSeed()

Overzicht les 2
vertakt programmaverloop

Nieuw data-type:
- ARRAY[]

Voorwaardelijke sprongen
- IF en GOTO
- SWITCH….CASE
- DO….WHILE

Herhaallussen
- FOR
- WHILE()

Hardware Libraries

Nieuwe functies
- serieel en I2C
- tone() en noTone()

Arrays

Twee problemen: looplicht (*vb. sketch 21*)

Twee oplossingen:

– een nieuw data-type: ARRAY[] gebruiken

– een FOR-lus gebruiken

Een array is een verzameling geheugenlocaties met data van hetzelfde type

Declaratie: **type naam_array[grootte]** Vb: **byte ledPin[5]**

grootte = aantal elementen in de verzameling – 1

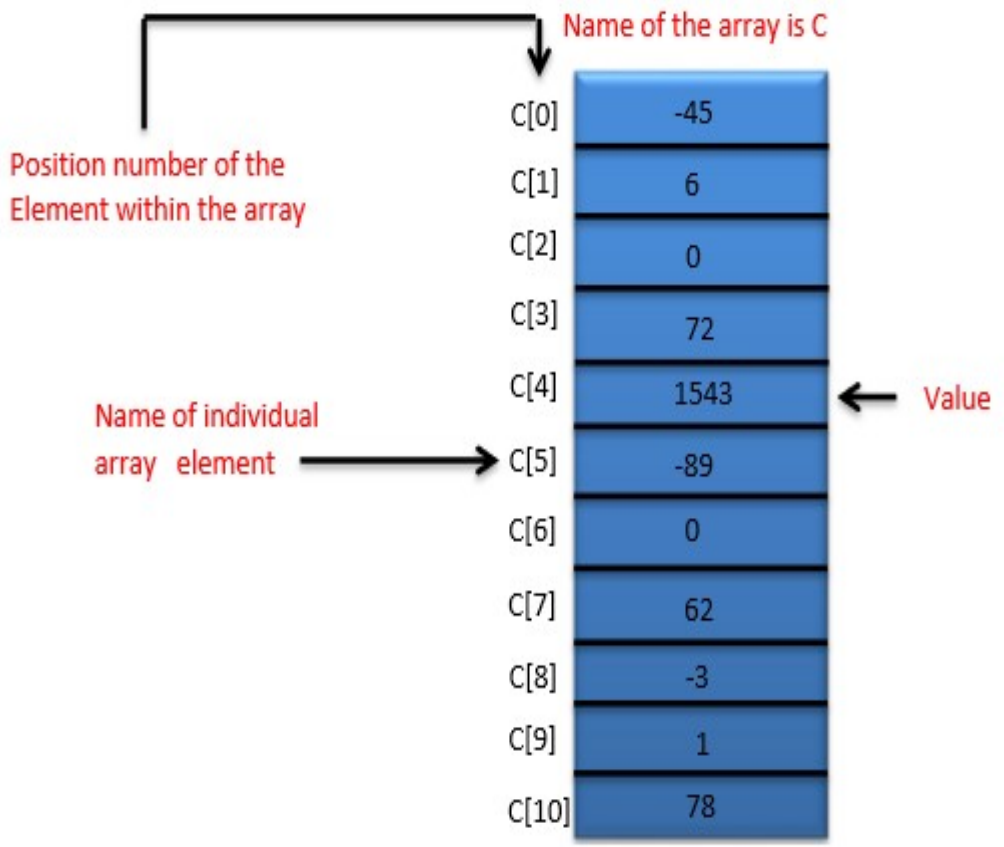
de elementen zijn ledPin[0], ledPin[1], ledPin[2]...ledPin[5]

na de declaratie hebben de elementen de waarde 0

Vullen van de elementen door opsomming van de elementen:

byte ledPin[5] = **{**4,5,6,7,8,9**}**

voorbeeld: **int C[10]:**



Kan ook anders!

met de FOR-lus (1)

Opvullen met een FOR-lus (alleen bij opeenvolgende waarden!):

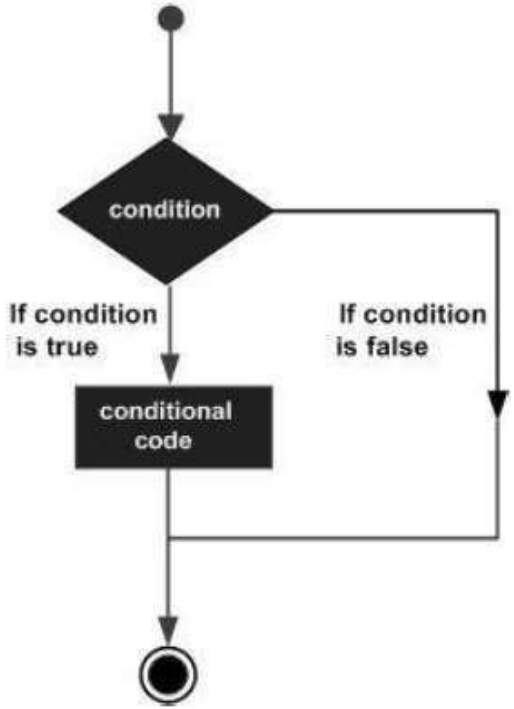
```
FOR( x=0, x=5, x++) {
    ledPin[x] = x + 4                    \\rechte haken!!!
}
```

En nu de oplossing… (sketch 21)

1. voorwaardelijke sprong: IF...

```
IF (expressie)
    opdracht;
```

```
IF (expressie)
   {
     opdrachtblok;
    }
```



Vb

```
IF (A>B);
   A++
```

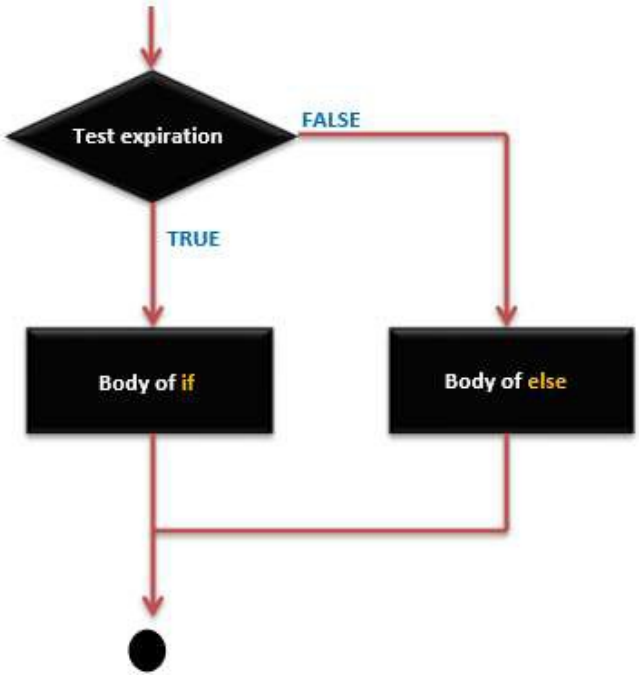
```
IF (A>B);
  {  A++;
     B--;  }
```

Voorbeeld
aan/uit van de L-led

```
int knopPin = 12;
int ledPin = 13;
int knopStatus = LOW;

void setup() {
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT);
    pinMode(knopPin, INPUT);
}

void loop() {

    knopStatus = digitalRead(knopPin);
    Serial.println(knopStatus);
    if (knopStatus == HIGH) {
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }

}
```

IF...ELSE sprongen

```
IF (expressie);
    { opdrachtblok1;}
ELSE
     {opdrachtblok2;}
```

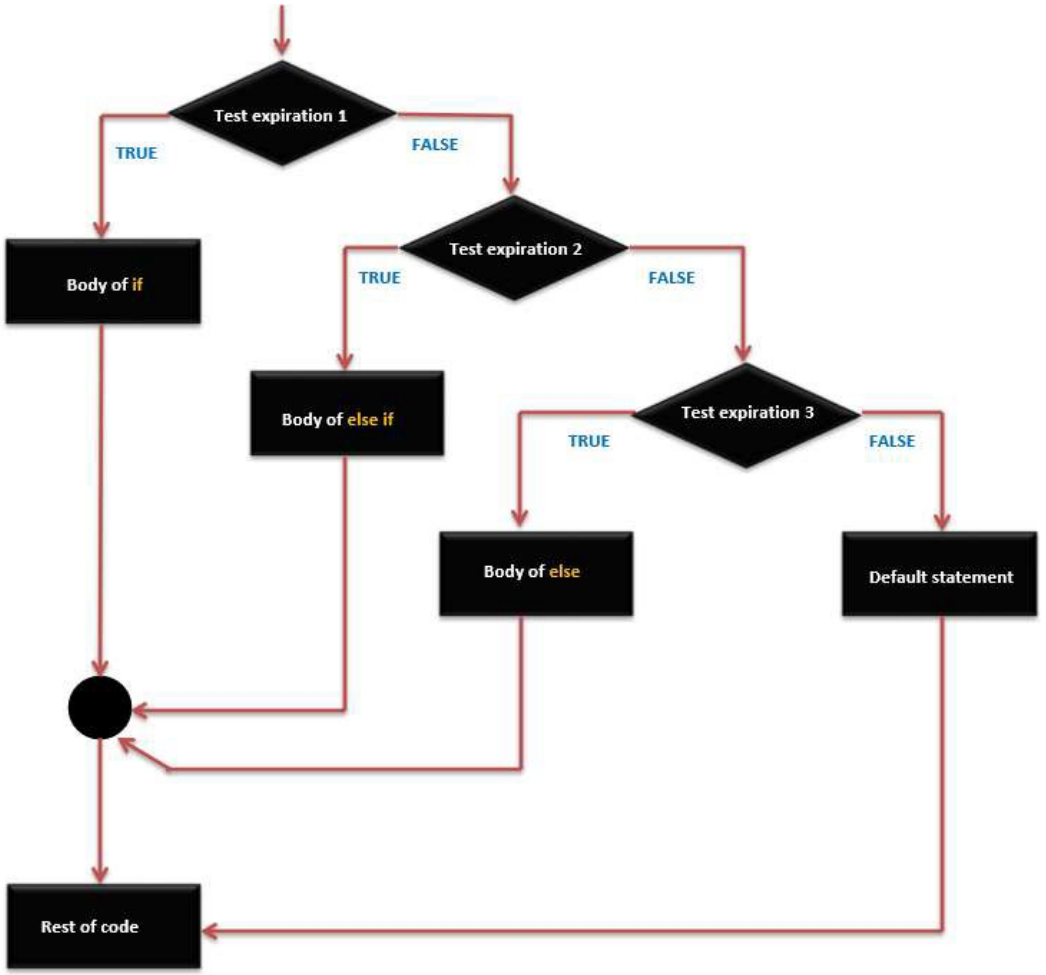


Vb

```
IF (spanning>4);
      { rodelamp = 1;
        groenelamp = 0; }
ELSE
      {rodelamp = 0;
       groenelamp = 1; }
```

Geneste IF...ELSE sprongen

C# laat geneste IF...ELSE sprongen toe:



Voorbeeld

```
int A=5;
int B=9;
int C=15;

void setup()
{}

void loop()
{
if A>B {A++}
      else if ((A==B)||(B>C) { C=B*A)}
      else C++;
}
```

Onvoorwaardelijke GOTO

- om tijdens het programmaverloop naar een gelabeld punt te springen
- **goto *label*;**
- ***Label* =** naam gevolgd door '**:**'

```
Vb:             .....
                goto jan;
                .....
                .....
          jan:
                .....
                .....
```

Het gebruik van een onvoorwaardelijke sprong wordt afgeraden...

maar kan soms een uitkomst bieden!

2. SWITCH...CASE – sprongen

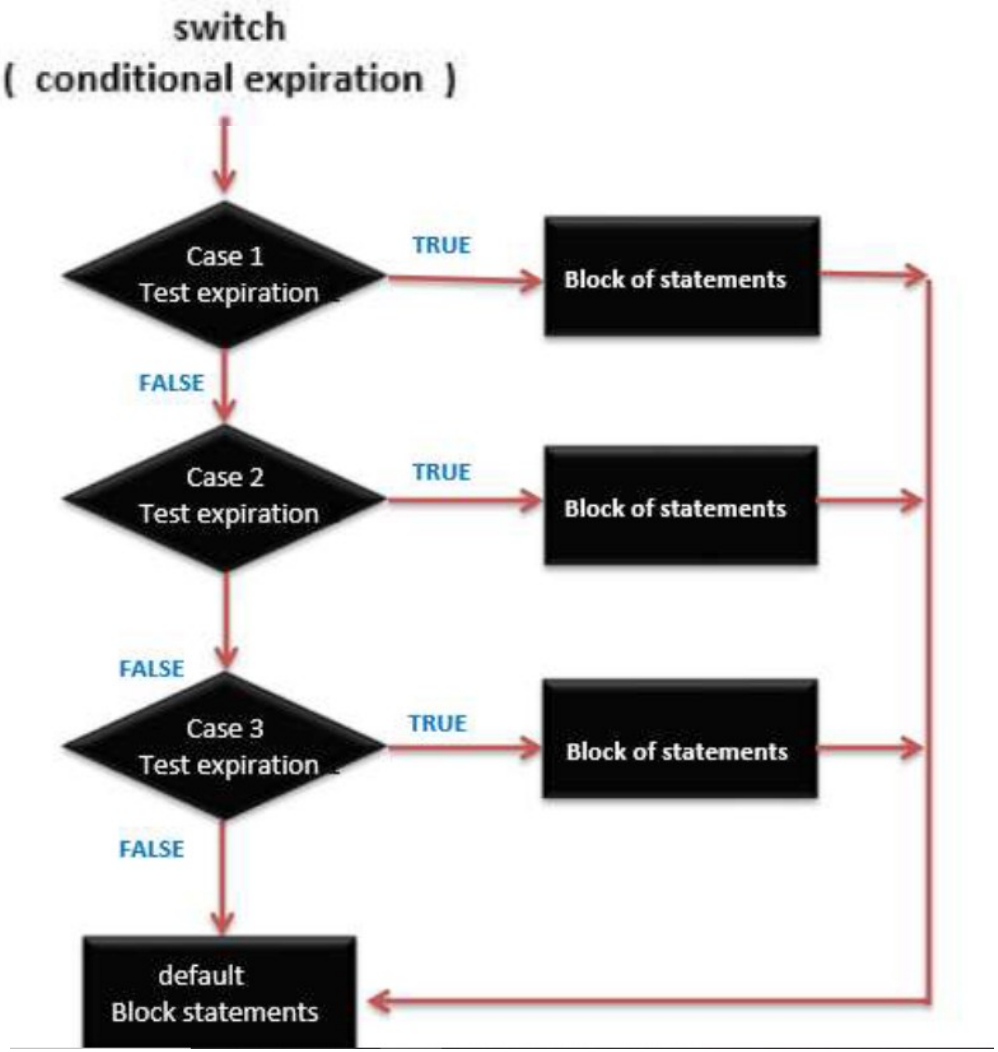
Syntax:

switch (variabele)

{

case 1: {…… break;}
case 2: {……} break;}

default: {……}



Voorbeeld:

Een variabele toestand kan drie waarden aannemen: 0, 1 ,2. Afhankelijk van de waarde moet iets verschillend uitgevoerd worden.

```
switch (toestand)

{
      case 0: low(); break;

      case 1: mid(); break;

      case 2: High(); break

      default: println(“ verboden toestand opgetreden!”);
}
```

Voorbeeld: 3 drukknoppen: knop1, knop2, knop3

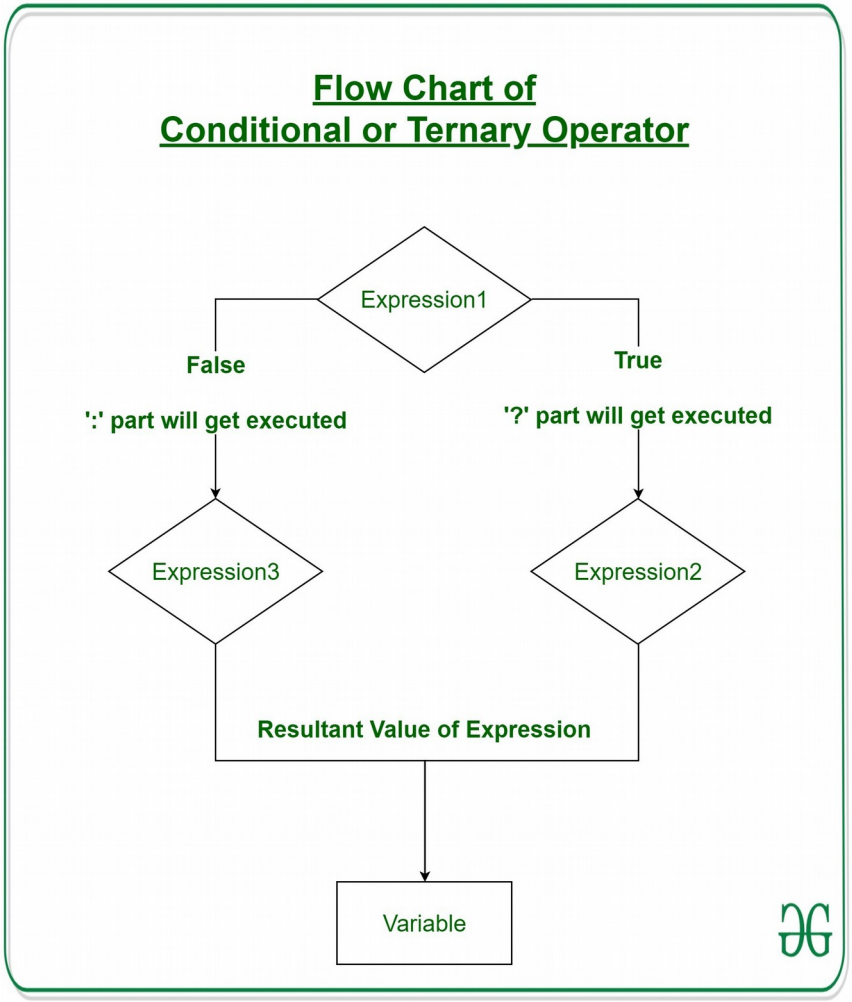
Afhankelijk van de gekozen knop moet een rode, groen of blauwe lamp branden

variabele knop kan 3 waarden aannemen: 1, 2, of 3

```
      …….
switch (knop)
{
  case knop1:
    {DigitalWrite(10,HIGH);}
    break;
 case knop2:
    {DigitalWrite(11,HIGH);}
    break;
  case knop3:
    {DigitalWrite(12,HIGH);}
    break;
default:
    // statements
 }
```

Conditionele (ternaire) operator **?**

- Bestaat uit 3 expressies
- Syntax: expressie1 **?** expressie 2 **:** expressie 3
- Als 1 waar is wordt 2 uitgevoerd en 3 genegeerd
- Als 1 onwaar is wordt 2 genegeerd en 3 uitgevoerd
- Het resultaat is **of** het resultaat van 2 **of** het resultaat van 3



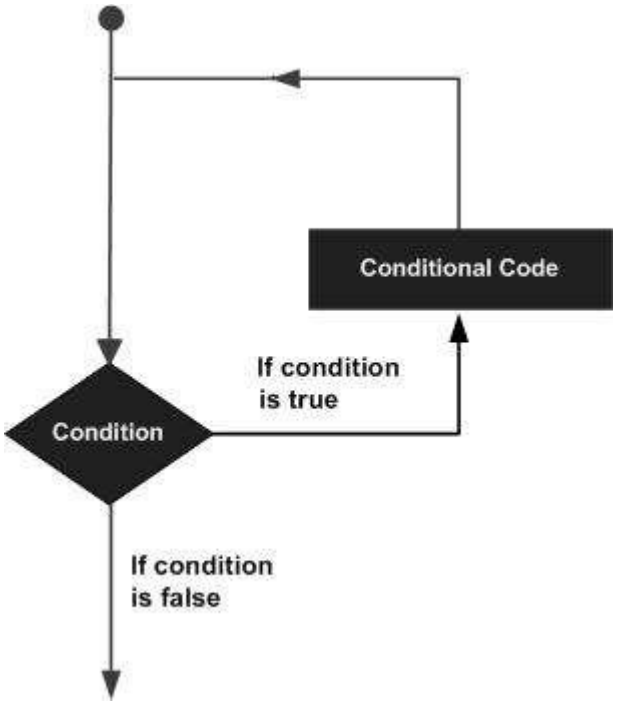
Voorbeeld: Even of oneven?

```
byte getal

void setup()
Serial.begin(9600);

Void loop()
Serial.print("Geef een getal");

If (Serial.available()) {
      getal=Serial.read();

      getal%2==0) ? Serial.println("even") :

      Serial.println("oneven")
}
return 0;
```

Lussen (loops)

Een loop dient om (een blok opdrachten) meerdere keren uit te voeren.

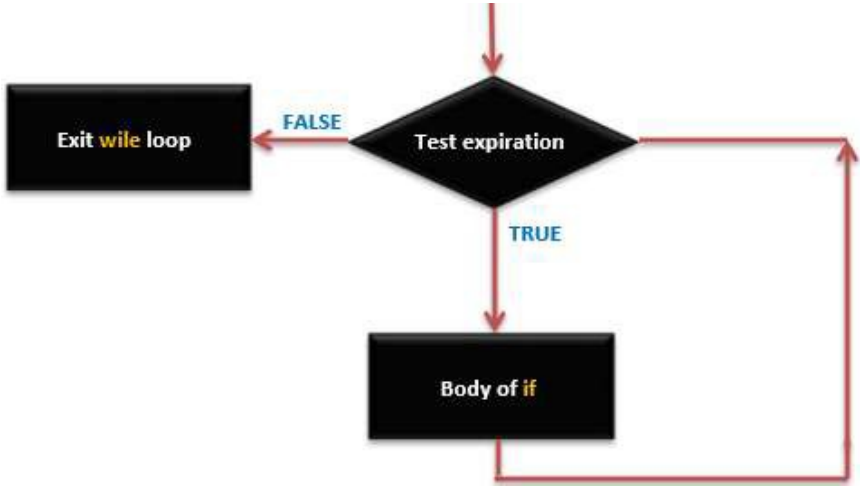
Algemene vorm:



soorten:

1. **WHILE loop**
2. **DO...WHILE loop**
3. **FOR loop**
4. **Geneste loop**
5. **Oneindige loop**

1. WHILE LOOP



- Wordt blijvend doorlopen tot de test (expressie) FALSE oplevert
- Syntax:

```
.....

while (expressie)
  {
     opdrachtblok
  }
.....
```

Voorbeeld

```
int x=0;
x=digitalRead(knop);
long count=0

while x==HIGH
{
      count++;
}

Serial.print(“het duurde “);
Serial print(count);
Serial.println(“tikken”)
```

2. DO...WHILE loop

Lijkt sterk op de while-loop, maar:

bij while-loop wordt voorwaarde getest voor uitvoering loop

bij do...while wordt de voorwaarde getest na uitvoering loop

Een do...while-loop wordt altijd tenminste één keer uitgevoerd!

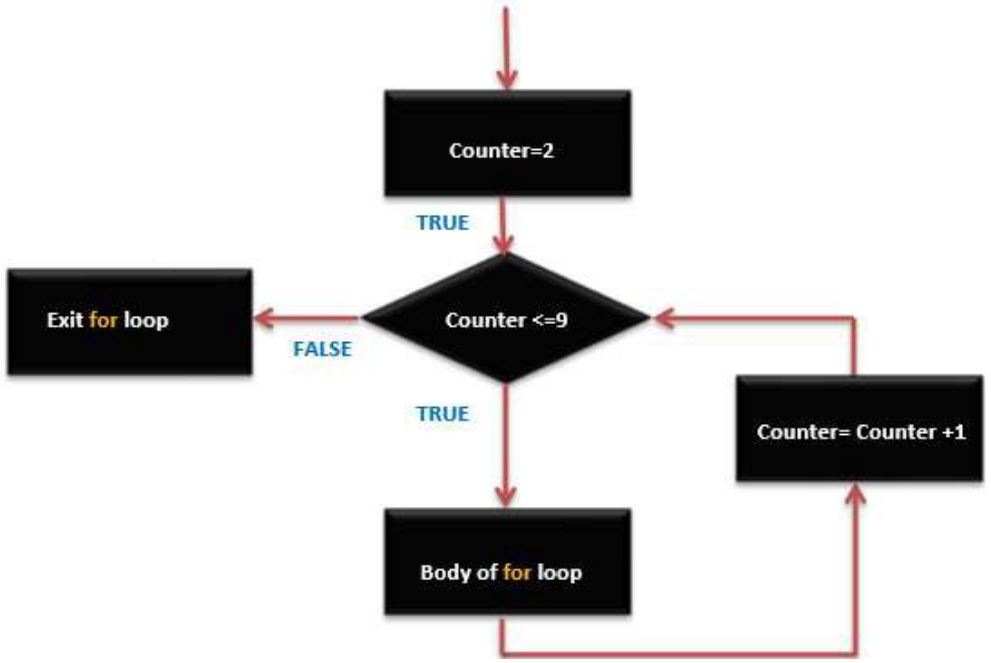
Syntax:

```
do
    { opdrachtenblok}
while
    (expressie)
```

3.FOR-loop (2)

Syntax

```
for(x=2; x<= 9; x++)
{
    opdrachtenblok
 }
```



FOR(beginwaarde teller; eindwaarde-expressie; verhoging/verlaging teller)

Voorbeeld een led dimmen:

```
int PWMpin = 10;

void setup() {
  // geen setup nodig
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(PWMpin, i);
    delay(200);
  }

 for (int i = 255; i >= 0; i--) {
    analogWrite(PWMpin, i);
    delay(200);
  }
}
```

4. Geneste loop

C# laat een loop in een loop toe

Voorbeeld:

```
for(i=0; i<=9; i++)
    {opdrachtenblok1
        for(j=0; j<=99; j++)
            { opdrachtenblok2}
        }
```

5. Oneindige loop

```
For(;;)
        { Pauze…}


While (1)
        { Pauze…}

do
        { Pauze...}
while(1)
```

```
Loop()
        { Pauze...)
```

Nieuwe functies: serieel en I2C

- Elke arduino heeft een hardware seriele poort
- Bij uno/nano zowel de USB(via FTDI-chip) als D0/D1 (=RX/DX)
- Protocol: 8 databit, 1 startbit, 1 stopbit
- Er is een buffer van 128 bytes
- Data kan verstuurd worden als ASCII-tekens (32 tot 127) of als numerieke waarde
- Controletekens: BS \b ;TAB \t; LF \n; CR \r vb: Serial.print('\t')
- Baudrate: 300...115200 bps

Seriele functie-library (Serial)

Automatisch meegeladen bij start Arduino

Serial.begin(*baudrate*)

stelt de baudrate in

op te nemen in SETUP()

Serial.available()

zonder argument; retourneert het aantal tekens in de buffer

Serial.read()

zonder argument; retourneert de eerste beschikbare byte in de buffer

Let op! Getal 1 wordt verstuurd als ASCII-teken 49!

```
vb  byte inkomendTeken = Serial.read();
```

Gebruik niet type CHAR, want dan blijft het een teken!

Om er terug een getal ervan te maken:

```
byte inkomendTeken = Serial.read()
inkomendTeken= inkomendTeken-48
```

of `inkomendTeken= inkomendTeken-'0'`

We lezen teken per teken! Getal 149 wordt verstuurd als 3 ASCII-tekens!

Serial.print(*argument*)

voor het versturen van data naar het serieel apparaat

vb: Serial.print(33)	33
Serial.print(3.14159)	3.15
Serial.print('A')	A
Serial.print(3.1459,4)	3.1459
Serial.print("dag jan")	dag jan

Serial.println(*argument*)

zelfde als Serial.print() + CR

Serial.write(*argument*)

zelfde als Serial.print() maar Serial.print(33) wordt geinterpreteerd als ASCII-letterteken (**=** !)

Kan maar telkens één byte versturen

SoftwareSerial library

- Om een seriele RX/TX te emuleren op elke I/O-pin
- Er is geen databuffer, maar werkt met interrupts
- Moet geladen worden als elke andere library

SYNTAX

SoftwareSerial *naam(rx-pin, tx-pin)*

vb: SoftwareSerial barcode(2,3);

Alle functies uit Serial.x kunnen gebruikt worden, maar voorafgegaan door *naam:*

barcode.begin()
barcode.read()
barcode.write() etc...

I2C-communicatie (wire library)

Principe: zie sheets opfrisavond digitale kennis

Voor I2C-communicatie wordt de library wire.h gebruikt:

#include <wire.h> (voor setup() plaatsen!)

De library omvat de functies:

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- available()
- write()
- read()
- SetClock()
- onReceiver()
- onRequest()

Let op hoofdletter!

wire.begin(adres);

om de I2C-bus klaar te maken

adres is niet nodig voor de master-arduino, wel als de arduino als slave gebruikt wordt

wire.beginTransmission(adres); (1)

om de slave aan te duiden waarmee gecommuniceerd wordt

wire.endTransmission(adres); (2)

om de communicatie met de verbonden slave af te sluiten

voordat een nieuwe slave aktief kan worden moeten (1) en (2) uitgevoerd zijn

wire.write(argument1,argument2);

om data te versturen van master naar slave.

argument1 moet type byte, type char of typr string zijn

argument2 is optioneel; is het aantal te versturen bytes

wire.requestFrom(*argument1, argument2***);**

verzoek om data te verzenden door een busapparaat
argument 1 is het adres; argument2 is het aantal te leveren bytes

wire.read();

heeft geen argument
leest de eerstvolgende afgeleverde byte

wire.available();

wordt door de master verstuurd naar de slave na een wire.requestFrom()
geen argument; retourneert het aantal te versturen bytes

wire.SetClock(*argument***);**

argument is 10000, 100000, 400000, 1000000. Meestal 100000
bepaalt de frequentie van de klok (SCL)

wire.onReceive(functie);

geeft de functie aan die opgeroepen moet worden als de slave data ontvangt van de master
retourneert niets

wire.onRequest(functie);

geeft de functie aan die opgeroepen moet worden als de master data ontvangt van de slave
retourneert niets

Nieuwe functies: tone() en notone()

tone(*pin, frequentie, ev. tijdsduur***)**

- om een blokspanning te genereren (50% duty cycle)
- genereert blijvend tenzij een tijdsduur in ms opgegeven wordt
- slechts mogelijk op één pin
- frequentiegebied: 31 Hz – 65535 Hz
- retourneert niets

Voorbeelden

tone(7, 230); blokspanning van 230 Hz op pin 7

tone(7, 230, 100); blokspanning van 230 Hz op pin7 gedurende 0,1 s

noTone(*pin*)

- stopt het genereren van een blokspanning (als met tone() geen tijdsduur ingesteld is
- retourneert niets
- NoTone() is ook nodig op van pin te wisselen!

Voorbeeld

noTone(7)

En toen kwam er een varken...