1. Datatyper

Datatyper er utbredt for alle programmeringsspråk og er absolutt ikke særegne for Arduino. Datamaskinen man programmerer på behøver informasjon om hvilken type data vi snakker om. Vi skal nå se på de grunnleggende datatypene.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datatype** | **Forklaring** |  | **Eksempel** |
| **Int** | Står for integer, og betyr heltall. |  | int x = 7; |
| **Float** | Desimaltall. |  | float x = 7.4; |
| **Char** | Står for character, og brukes for å lagre bokstaver. Kan også lagre ord og setninger |  | char x = "a";  char x[] = "hei"; |
| **Bool** | Står for boolean og lagrer enten 1 eller 0. |  | bool x = 0; |

Dette er altså de mest grunnleggende datatypene vi har, men det finnes mange flere.

Over har vi sett på hvordan datatyper kan brukes på variabler. En variabel er et egendefinert område i minnet på maskinen hvor du har lagt informasjon du ønsker å ta vare på. Enten at x er 5, eller et array som inneholder alle bursdagene til klassekameratene dine. Funksjoner har også datatype, dette kommer vi tilbake til under kapittelet om funksjoner.

1. Arrays

Den observante leser har kanskje merket seg noe spesielt med den ene char-variabelen rett over. Den har noen merkelige tegn bak seg (‘[‘ og ‘]’). Dette er det vi kaller et array, en er en rekke med verdier. Følgende eksempel viser forskjellige metoder å deklarere et array på, både med og uten verdier.

|  |
| --- |
| Eksempel på if/else |
| int bursdager**[]** **=** **{**1**,** 5**,** 7**,** 9**,** 10**};**  int bursdag\_per **=** bursdager**[**0**];**  int bursdag\_ane **=** bursdager**[**3**];**  char navn**[]** **=** "thomas"**;**  void setup**()** **{**  Serial**.**begin**(**9600**);**  **}**  void loop**()** **{**  Serial**.**println**(**bursdager**[**3**]);**  Serial**.**println**(**navn**[**2**]);**  Serial**.**println**(**""**);**    **for(**int i **=** 0**;** i **<** 5**;** i**++)** **{**  Serial**.**print**(**bursdager**[**i**]);**  Serial**.**print**(**" "**);**  **}**  Serial**.**println**(**""**);**  **for(**int i **=** 0**;** i **<** 6**;** i**++)** **{**  Serial**.**print**(**navn**[**i**]);**  Serial**.**print**(**" "**);**  **}**  Serial**.**println**(**""**);**  Serial**.**println**(**""**);**  delay**(**2000**);**  **}** |

For å hente ut enkeltverdier benytter man seg av noe som kalles indeksering. Her er det viktig å innse at det første elementet i et array, er indeks null!

# 

1. If- og Else-strukturen

Se for deg en lyssensor. Vi kopler den på vår Arduino, skrur av alle lamper i rommet og leser av en lav verdi. Der etter skrur vi lampene på, og leser den nye verdien. Den skal være høy, der som alt er koplet riktig. På hvilken måte kan vi nå få vår Arduino til å gjøre en handling når rommet er mørkt, og en annen når rommet er lyst? Her kommer if og else inn i bildet. De gjør akkurat det de høres ut som, og bruken av de kan se slik ut.

|  |
| --- |
| Eksempel på if/else |
| **void loop(){**  **int x = 3;**  **int y = 0;**  **if (x == 3) {**  **Serial.println("X er 3");**  **}**  **else {**  **Serial.println("X er noe annet");**  **}**    **if (y >= 1) {**  **Serial.println("Y er større eller lik 1");**  **}**  **else {**  **Serial.println("Y er mindre enn 1");**  **}**  **}** |

Dette eksempelet vil resultere i en tekst på serie-monitoren, avhengig av verdien til x. Parentesene bak if er det som kalles argumentet til strukturen. Merk for øvrig de to likhetstegnene. Disse kalles sammenliknende operatorer, og er til for å sjekke en verdi i forhold til en annen. Det finnes flere av disse, og kan leses om ved et raskt google-søk på “Comparison Operators C++.”

Hvis vi hadde kjørt følgende kode, ville serie-monitoren skrevet ut “X er 3”, etterfulgt av “Y er 0 eller mindre” for hver gjennomgang av **void loop().** Husk at denne løkken gjentar seg selv så lenge Arduino har strøm.

1. While- og For-løkkene

Hva om vi ønsket at programkoden ikke skulle gå videre før x var noe annet enn 3? Dette kunne vi løst med en while-løkke. En while-løkke vil “sitte fast” i arbeidsoppgaven sin, helt til argumentet ikke lengre stemmer. En while-løkke har ett argument, og vil kjøre så lenge dette argumentet er sant. Dette kan vi se litt nærmere på.

|  |
| --- |
| Eksempel på while-løkke |
| int x **=** 0**;**  void loop**(){**  **while(**x **<=** 10**)** **{**  Serial**.**print**(**"X er mindre enn 10: "**);**  Serial**.**println**(**x**);**  x **=** x **+** 1**;**  **}**  Serial**.**println**(**"X er over, eller lik 10"**);**  Serial**.**println**(**"Resetter X-verdi"**);**  x **=** 0**;**  **}** |

Hvis du kjører denne koden, vil seriemonitoren skrive ut «X er mindre enn 10» ti ganger, etterfulgt av verdien til x. Merk at x begynner på null, og vil stige helt til ni. Etter dette vil x bli ti, argumentet vil ikke lengre stemme, og koden i while-løkken blir hoppet over.

Måten vi brukte while-løkken på nå, minner veldig om en annen løkke-struktur vi har. For-løkken. Det er en løkke som brukes for å telle, enten opp eller ned. For-løkken krever et litt spesielt argument, som ser tre-delt ut. Kort forklart er disse tre argumentene startverdi, sluttverdi og inkrement. Inkrement betyr hvor mye vi skal telle. Dette kan forklares med en liten pseudo-kode.

|  |
| --- |
| **for(**startverdi**;** så lenge dette stemmer**;** med følgende inkrement**)** **{**  //Kode  **}** |

La oss se på hvordan dette faktisk skal se ut. Vi kan bygge opp en for-løkke som teller fra tre til ni på denne måten.

|  |
| --- |
| Eksempel på for-løkke |
| **for(**int i **=** 3**;** i **<=** 9**;** i**++)** **{** //PS: i++ er ekvivalent med i = i + 1.  Serial**.**println**(**i**);**  **}** |

Vi kunne selvsagt brukt hvilket som helst navn på vår tellervariabel i, men det er standard praksis å bruke i på enkle for-løkker. Hva skjer om vi nøster enda en for-løkke inni denne? Å forstå hva som skjer da, vil gi en god forståelse for hvordan flyten i koden din fungerer. Kopier følgende kode, studer den, last den opp og se hva seriemonitoren sier.

|  |
| --- |
| Eksempel på nøstet for-løkke |
| void setup**()** **{**  Serial**.**begin**(**9600**);**  **}**  void loop**()** **{**  **for** **(**int x **=** 0**;** x **<** 9**;** x**++)** **{**  Serial**.**print**(**"X er: "**);**  Serial**.**println**(**x**);**  **for** **(**int y **=** 0**;** y **<** 9**;** y**++)** **{**  Serial**.**print**(**"Y er: "**);**  Serial**.**println**(**y**);**  **}**  **}**  **}** |

Merk hvordan den indre løkken teller seg ferdig, før den ytre løkken inkrementerer. Dette må bety når en for-løkke er påbegynt, vil Arduino sitte fast i denne løkken, helt til den er ferdigtelt. Men det gjenstår fortsatt et par runder i den ytre for-løkken.

1. Switch-case

En switch-case kan sammenliknes med en rekke if-statements. Se for deg følgende eksempel.

|  |
| --- |
| Eksempel på if-statements |
| int x = 0**;**  **if** **(**x == 0**)** **{**  Serial**.**println**(**"x = 0"**);**  **}**  **if** **(**x == 1**)** **{**  Serial**.**println**(**"x = 1"**);**  **}**  **if** **(**x == 2**)** **{**  Serial**.**println**(**"x = 2"**);**  **}**  **if** **(**x == 3**)** **{**  Serial**.**println**(**"x = 3"**);**  **}** |

Her kan man raskt se at dersom x kan ha ganske mange verdier, kan dette fort bli rotete. En switch-case kan hjelpe oss med akkurat dette. Alt vi må gjøre er å fortelle switch-casen hvilken variabel den skal “switche” på, og hvilke verdier vi bryr oss om. For eksempel slik.

|  |
| --- |
| Eksempel på switch-case |
| int x = 0**;**  **switch(**results**.**value**)** **{**  **case** 0**:**  Serial**.**println**(**"x = 0"**);**  **break;**  **case** 1**:**  Serial**.**println**(**"x = 1"**);**  **break;**  **case** 2**:**  Serial**.**println**(**"x = 2"**);**  **break;**  **case** 3**:**  Serial**.**println**(**"x = 3"**);**  **break;**  **}** |

La oss nå kombinere dette med en egenbygd funksjon, slik at vi kan få en litt ryddigere “main-sløyfe”.

|  |
| --- |
| Eksempel switch-case i egen funksjon |
| **int x = 0;**  **int y = 2;**    **void setup() {**  **Serial.begin(9600);**  **}**    **void loop() {**  **switchFunksjon(x);**  **delay(100);**  **switchFunksjon(y);**  **delay(100);**  **}**    **void switchFunksjon(argument) {**  **switch(argument) {**  **case 0:**  **Serial.println("argumentet er 0");**  **break;**  **case 1:**  **Serial.println("argumentet er 1");**  **break;**  **case 2:**  **Serial.println("argumentet er 2");**  **break;**  **case 3:**  **Serial.println("argumentet er 3");**  **break;**  **}**  **}** |

# 

# 

1. Funksjoner

Funksjoner er en viktig del av programmering. De rydder opp kode, fjerner behovet for å gjenta liknende kode og tillater modularitet i forhold til arbeidsoppgave. Dette skal vi se nærmere på her.

|  |  |
| --- | --- |
| Eksempel på gjentakene kode | |
| int x **=** 0**;**  void loop**()** **{**  **if** **(**x **==** 0**)** **{**  delay**(**100**);**  **}**  **else** **if** **(**x **==** 1**)** **{**  digitalWrite**(**ledPin**,** HIGH**);**  delay**(**100**);**  digitalWrite**(**ledPin**,** LOW**);**  delay**(**100**);**  **}**  **else** **{**  digitalWrite**(**ledPin**,** HIGH**);**  delay**(**100**);**  digitalWrite**(**ledPin**,** LOW**);**  delay**(**100**);**  **}**  **}** | int x **=** 0**;**  void loop**()** **{**  **if** **(**x **==** 0**)** **{**  delay**(**100**);**  **}**  **else** **if** **(**x **==** 1**)** **{**  flashLed**();**  **}**  **else** **{**  flashLed**();**  **}**  **}**  void flashLed**()** **{**  digitalWrite**(**ledPin**,** HIGH**);**  delay**(**100**);**  digitalWrite**(**ledPin**,** LOW**);**  delay**(**100**);**  **}** |

Nå kan vi altså enkelt skrive flashLed() når vi ønsker å blinke dette lyset. Merk at funksjonen flashLed() blir skrevet *etter* loop().

Men hva om vi ønsker å bruke den samme funksjonen på forskjellige pinner? Det er her funksjoner virkelig viser sin sterke side. Vi kan sende argumenter inn i en funksjon, for deretter å bruke de. La oss se på dette.

|  |
| --- |
| Eksempel på argumenter |
| **int x = 0;**  **int ledPin1 = 12;**  **int ledPin2 = 13;**  **int delay1 = 100;**  **int delay2 = 200;**    **void loop() {**  **if (x == 0) {**  **delay(100);**  **}**  **else if (x == 1) {**  **flashLed(ledPin1, delay1);**  **}**  **else {**  **flashLed(ledPin2, delay2);**  **}**  **}**  **void flashLed(int pinNumber, int delayTime) {**  **digitalWrite(pinNumber, HIGH);**  **delay(delayTime);**  **digitalWrite(pinNumber, LOW);**  **delay(delayTime);**  **}** |

En annen viktig komponent av funskjoner er evnen til å returnere en verdi. Du har kanskje sett ordet void foran både loop, setup og flashLed, og lurt på hva dette betyr? Det er datatypen denne funksjonen har. Dette er knyttet sammen med hvilke type verdier funksjonen kan returnere. Void betyr tomrom, og inneholder absolutt ingen verdi. Det kan sees på som den matematiske tomme mengde. En funksjon kan ha alle datatyper en normal variabel kan ha. La oss se på hvordan vi kan utnytte returnering av verdier.

|  |
| --- |
| Eksempel på argumenter |
| void loop**()** **{**  int tall1 **=** 3**;**  int tall2 **=** 7**;**    int c **=** summer**(**tall1**,** tall2**);**  int d **=** tall1 **+** tall2**;**    Serial**.**print**(**c**);**  Serial**.**print**(**" = "**);**  Serial**.**print**(**d**);**  **}**    int summer**(**int a**,** int b**)** **{**  **return** **(**a **+** b**);**  **}** |

Merk her hvordan summer(tall1, tall2) gir samme verdi som tall1 + tall2. Dette er ikke tilfeldig. Den returnerte verdien til funksjonen, blir altså “lagt i” ordet summer(tall1, tall2). Dette kan du også se ved å for eksempel skrive Serial.println(summer(tall1, tall2));