**FUNGSI DAN PERINTAH YANG DIJALANKAN ARDUINO**

1. **STRUCTURE**
2. setup()

berfungsi ketika sketsa dimulai

1. loop()

menginisialisasi dan menetapkan nilai awal, untuk melakukan perulangan pada program.

**Control Structure**

1. If

Sebagai operator perbandingan untuk mencapai suatu kondisi tertentu.

1. If.. else

Untuk control yang mempunyai lebih dari satu kondisi.

1. for

Digunakan untuk peningkatan dan mengakhiri loop. Berguna untuk setiap operasi berulang-ulang, dan sering digunakan dalam kombinasi dengan array untuk beroperasi pada koleksi data / pin.

1. Switch case
2. Mengontrol aliran program dengan memungkinkan programmer untuk menentukan kode yang berbeda yang harus dilaksanakan dalam berbagai kondisi. membandingkan nilai variabel dengan nilai-nilai yang ditentukan jika ditemukan kecocokan maka dijalakan.
3. While

Selama statement bernilai TRUE maka akan terus melakukan perulangan perintah.

1. Do..while

Hampir sama dengan pernyataan “while”, bedanya bila pada pernyataan “while” , test kondisi diuji dahulu, dan bila test kondisi bernilai benar maka pernyataan yang ada di dalam blok “while” akan dieksekusi, kemudian test kondisi akan diinkremental. Pada pernyataan “do-while”, kondisi menjadi terbalik, yaitu pernyataan utama akan dieksekusi terlebih dahulu, setelah itu baru test kondisi diuji, jika test kondisi benar maka pernyataan utama akan diulang, dan jika salah program akan keluar dari blok “do-while”.

1. Break

Digunakan untuk keluar dari pernyataan **do** , atau **while loop**, melewati kondisi loop normal. Hal ini juga digunakan untuk keluar dari pernyataan **switch**.

1. Continue

Pernyataan ini melewatkan sisa dari iterasi saat ini loop (lakukan, untuk, atau saat). Dengan memeriksa ekspresi kondisional dari loop, dan melanjutkan dengan iterasi berikutnya.

1. Return  
   Untuk menghentikan fungsi dan mengembalikan nilai dari fungsi ke fungsi panggilan, jika diinginkan
2. Goto

Instruksi untuk mengarahkan eksekusi program ke pernyataan yang diawali oleh sebuah label atau identifier khusus seperti tanda (:) lalu mengarahkan eksekusi program **tampil** yang diikuti dengan mengerjakan perintah yang sudah ditaruh pada label. Perintah goto ini juga bisa digunakan untuk keluar dari *loop* dan *nested loop* dengan lebih mudah dan ringkas.

**Further Syntax**

1. ; (semicolon)

Digunakan setelah berakhirnya *statement*. Merupakan suatu pembatas disetiap *statement*.

1. {} (curly braces)

Mendefinisikan awal dan akhir dari sebuah blok fungsi

1. // (single line comment)

Sama dengan *multi-line coment* hanya saja perbaris.

1. /\*\*/ (multi-line comment)

Semua *statement* yang di tulis dalam block comments tidak akan di eksekusi dan tidak akan di compile sehingga tidak mempengaruhi besar program yang di buat untuk di masukan dalam board arduino

1. #define

Komponen yang digunakan untuk memberi nama pada nilai konstanta sebelum program dikompilasi

1. #include

Memasukkan sketsa program ke dalam *libraries*.

**Arithmetic Operator**

Operator yang digunakan untuk memanipulasi angka.

1. = (assignment operator)

Sama dengan (hasil)

1. + (addition)

Penambahan

1. – (subtraction)

Pengurangan

1. \* (multiplication)

Perkalian

1. / (division)

Pembagian

1. % (modulo)

Menghasilkan sisa dari hasil pembagian suatu angka dengan angka yang lain

**Comprasion Operator**

Operator yang digunakan untuk membandingkan nilai logika.

1. == (equal to)

Sama dengan (misalnya: 12 == 10 adalah FALSE (salah) atau 12 == 12 adalah TRUE (benar))

1. != (not equal to)

Tidak sama dengan (misalnya: 12 != 10 adalah TRUE (benar) atau 12 != 12 adalah FALSE (salah))

1. [<](https://www.arduino.cc/en/Reference/If) (less than)

Lebih kecil dari (misalnya: 12 < 10 adalah FALSE (salah) atau 12 < 12 adalah FALSE (salah) atau 12 < 14 adalah TRUE (benar))

1. [>](https://www.arduino.cc/en/Reference/If) (greater than)

Lebih besar dari (misalnya: 12 > 10 adalah TRUE (benar) atau 12 > 12 adalah FALSE (salah) atau 12 > 14 adalah FALSE (salah))

1. [<=](https://www.arduino.cc/en/Reference/If) (less than or equal to)

Menyatakan operand pertama lebih kecil atau lebih kecil sama dengan operand kedua.

1. [>=](https://www.arduino.cc/en/Reference/If) (greater than or equal to)

Menyatakan operand pertama lebih besar atau lebih besar sama dengan operand kedua.

**Boolean Operators**

Digunakan pada kondisi if dalam *statement*.

1. [&&](https://www.arduino.cc/en/Reference/Boolean) (and)

Ouput akan bernilai TRUE jika kedua operand bernilai TRUE.

1. [||](https://www.arduino.cc/en/Reference/Boolean) (or)

Output akan bernilai FALSE jika kedua operand bernilai FALSE.

1. [!](https://www.arduino.cc/en/Reference/Boolean) (not)

Operator NOT menyatakan kebalikan nya. Contoh jika input bernilai FALSE output akan bernilai TRUE.

**Pointer Access Operators**

Berfungsi sebagai penunjuk baik ke variable, fungsi atau objek.

1. [\* dereference operator](https://www.arduino.cc/en/Reference/Pointer)

Address of atau alamat dari variable/fungsi tersebut di dalam memory cel

1. [& reference operator](https://www.arduino.cc/en/Reference/Pointer)

Value pointed by atau nilai dari variable yang direferensikan.

**Bitwise Operators**

1. [&](https://www.arduino.cc/en/Reference/BitwiseAnd) (bitwise and)

Digunakan diantara dua ekpresi bilangan bulat yang dioperasikan pada perhitungan biner. Jika kedua masukan bit adalah 1, output yang dihasilkan adalah 1, jika output adalah 0.

1. [|](https://www.arduino.cc/en/Reference/BitwiseAnd) (bitwise or)

Digunakan diantara dua ekpresi bilangan bulat yang dioperasikan pada perhitungan biner. Output bernilai 1 kecuali jika input kedua bilangan biner adalah 0.

1. [^](https://www.arduino.cc/en/Reference/BitwiseAnd) (bitwise xor)

Operator ini mirip dengan bitwise operator OR kecuali jika bernilai 1 untuk posisi tertentu kapan tepatnya salah satu bit masukan untuk posisi yang 1. Jika keduanya 0 atau keduanya 1, operator XOR mengevaluasi ke 0

1. [~](https://www.arduino.cc/en/Reference/BitwiseXorNot) (bitwise not)

Tidak seperti & dan |, bitwise NOT operator diterapkan pada operan tunggal. Bitwise NOT berubah setiap bit untuk kebalikannya: 0 menjadi 1, dan 1 menjadi 0.

1. [<<](https://www.arduino.cc/en/Reference/Bitshift) (bitshift left)

Digunakan untuk operator pergeseran nilai biner ke kiri.

Contoh : int a = 00000111

int b = a << 3

int c = b << 3

artinya b = a digeser kekiri 3 bit atau b = 00111000

1. [>>](https://www.arduino.cc/en/Reference/Bitshift) (bitshift right)

Digunakan untuk operator pergeseran nilai biner ke kanan.

Contoh : int a = 00000111

int b = a >> 2

int c = b >> 3

artinya b = a digeser kekanan 2 bit atau b = 11000001

**Compound Operators**

1. [++](https://www.arduino.cc/en/Reference/Increment) (increment)

Kenaikan variable. Menambahkan suatu nilai variabel

1. [--](https://www.arduino.cc/en/Reference/Increment) (decrement)

Penurunan variable. Mengurangi suatu nilai variable.

Melakukan operasi matematika pada variabel dengan konstanta lain atau variabel.

1. [+=](https://www.arduino.cc/en/Reference/IncrementCompound) (compound addition)

Penjumlahan gabungan

1. [-=](https://www.arduino.cc/en/Reference/IncrementCompound) (compound subtraction)

Pengurangan gabungan

1. [\*=](https://www.arduino.cc/en/Reference/IncrementCompound) (compound multiplication)

Perkalian gabungan

1. [/=](https://www.arduino.cc/en/Reference/IncrementCompound) (compound division)

Pembagian gabungan

1. [%=](https://www.arduino.cc/en/Reference/IncrementCompound) (compound modulo)

Hasil bagi gabungan

1. [&=](https://www.arduino.cc/en/Reference/BitwiseCompoundAnd) (compound bitwise and)

sering digunakan dengan variabel dan konstanta untuk memaksa bit tertentu dalam variabel untuk negara LOW (dengan 0). Hal ini sering disebut dalam panduan pemrograman sebagai "clearing" atau "ulang" bit.

1. [|=](https://www.arduino.cc/en/Reference/BitwiseCompoundOr) (compound bitwise or)

Digunakan dengan variabel dan konstanta untuk "mengatur" (set ke 1) bit tertentu dalam variabel.