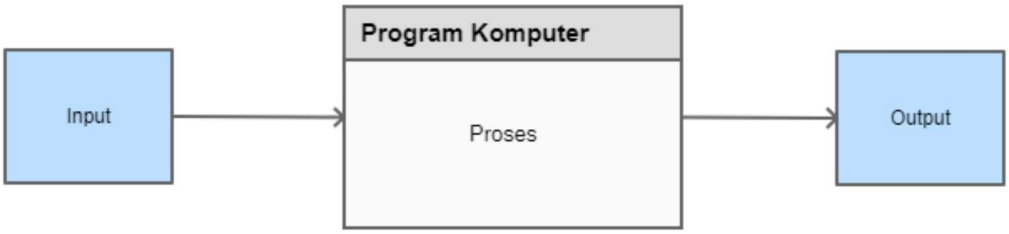
4.2 Praktikum Pemrograman Bahasa C Arduino Variable, Control Structure, Further Syntax

Indobot Academy 21 November 2022

1. Dasar Teori

Program komputer pada umumnya memiliki 3 bagian penyusun logika yang terdiri dari: input data (masukan), process data (pemrosesan), dan output data (keluaran).



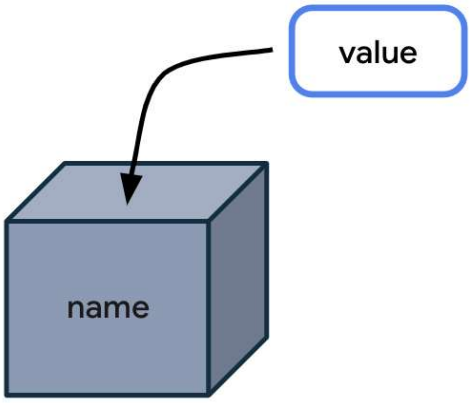
Gambar 1. Proses Kerja Komputer

Dalam pemrograman Arduino, nilai input bisa kita dapatkan dari sebuah sensor, button, dan sebagainya. Sementara untuk output dapat kita tampilkan melalui serial monitor, nyala led, display LED, buzzer, aktuator, dan masih banyak output lainnya.

Dalam tahap pemrosesan, program membutuhkan bantuan variabel untuk menyimpan data sementara. Sama halnya ketika kita berpikir, kita akan membutuhkan beberapa ingatan untuk memproses informasi.

1.1. Variabel

Variabel adalah sebuah tempat menyimpan sebuah nilai. Sementara tipe data adalah jenis nilai yang akan tersimpan dalam variabel. Variabel dapat diibaratkan seperti kotak yang kita beri nama dan didalamnya menyimpan benda tertentu. Jika diilustrasikan, variabel akan terlihat seperti gambar 2 berikut.



Gambar 2. Ilustrasi Variabel

Pada pelajaran matematika, kita sering menemukan `x` dan `y` .

Nah Si `x` dan `y` ini disebut variabel, karena tugasnya menyimpan nilai.

```
x = 3;
y = 4;
```

Masih kebingungan juga?

Anggap saja variabel itu sebuah wadah, lalu tipe data itu jenis-jenis benda yang akan disimpan dalam wadah tersebut.

1.1.1. Bagaimana Cara Membuat Variabel pada Bahasa Pemrograman C?

Berikut ini merupakan cara untuk membuat variabel pada bahasa Pemrograman C:

Gambar 3. Contoh Variabel

Jadi, pertama kita harus menuliskan tipe data lalu diikuti dengan nama variabelnya.

Seperti ini contohnya,

```
int tinggi;
```

Kita akan membuat variabel dengan nama `tinggi` dan tipe datanya adalah `int` *(integer)*. Jangan lupa, di setiap pembuatan variabel harus diakhiri dengan titik koma. Saat membuat variabel, kita juga dapat langsung mengisi dengan nilai yang akan kita simpan.

Contoh :

```
int berat = 49;
```

Ini berarti kita akan membuat variabel dengan nama `berat` dan tipe data *integer*, lalu langsung diisi dengan nilai `49` .

1.1.2. Aturan Penulisan Variabel pada C

Ada beberapa aturan dalam penulisan variabel yang harus kita ketahui :

1. Nama variabel tidak boleh didahului dengan simbol dan angka.
2. Nama variabel tidak boleh menggunakan kata kunci yang sudah ada pada bahasa C, contoh: `if` , `int` , `void` , dll.
3. Nama variabel bersifat *case sensitive*, artinya huruf besar dan kecil dibedakan, contoh: `nama` dan `Nama` adalah dua variabel yang berbeda.
4. Disarankan menggunakan *underscore* untuk nama variabel yang terdiri dari dua suku kata, jangan menggunakan spasi karena akan terjadi error ketika dijalankan. Contoh yang benar: `nama_saya` .

Contoh :

```
nama_saya = Fulan;
makanan_favorit = Sate;
```

1.2. Tipe Data Pada Bahasa C

Menurut Wikipedia:
“Tipe data atau kadang disingkat dengan ‘tipe’ saja adalah sebuah pengelompokan data untuk memberitahu *compiler* atau *interpreter* bagaimana programmer ingin mengolah data tersebut”.

Jenis Tipe Data	Penjelasan	Contoh Penulisan
Integer	Tipe data untuk angka bilangan bulat	`int variabel = 1,2,3,4,10,100;`
Float	Tipe data untuk angka bilangan desimal	`float variabel = 1.0, 20.1, 50.25;`
String	Tipe data untuk teks, seperti nama, kalimat, kata dan lain-lain	`String nama = "Fulan";`

Boolean	Tipe data untuk logika boolean (true dan false)	`bool variabel = true;` `bool variabel_2 = false;`
Char	Tipe data char adalah tipe data yang hanya terdiri dari satu karakter saja	`char jenis_kelamin = “P”;` `char golonganDarah = "O";`
Double	Tipe data double mirip seperti float hanya saja kapasitas penyimpannya lebih besar.	`double angka = 1.20;` `double angka_20 = 25.30;`

1.3. Mengenal Konstanta pada C

Konstanta merupakan bilangan tunggal yang nilainya tidak akan berubah (konstan). Penamaan variabel dan konstanta pada dasarnya tidak boleh sama persis dengan sintaks bawaan, misalnya: for, if, while dan sebagainya. Selain itu nama tidak boleh ada spasi, bila hendak memisahkan 2 buah kata untuk menyatakan variabel atau konstanta harus dihubungkan dengan tanda garis bawah (_) misalnya: “waktu_tunggu”.

Jika anda tidak mengikuti arahan di atas, maka akan ada potensi error pada saat compile. Terdapat 2 cara pembuatan konstanta dalam bahasa C :

- Menggunakan `#define`
- Menggunakan `const`

1.3.1. #define

Fungsi define pada Arduino yaitu memungkinkan programmer untuk menamai suatu bilangan konstan sebelum program dapat dikompilasi. Sebagai contoh, penggunaan `#define` biasanya digunakan untuk mendeklarasikan nomor pin perangkat. Selain itu juga dapat dipakai untuk keperluan IoT misalnya server. Hal tersebut dapat ditulis dengan format sebagai berikut :

```
#define ledPin 3
#define server "thingsboard.cloud"
```

Keterangan:

- ledPin dan server = variabel yang digunakan.
- 3 = nilai integer sebagai inisialisasi pin yang sedang dipakai, berada pada GPIO 3.
- thingsboard.cloud = nilai String sebagai inisialisasi platform IoT yang sedang dipakai.

1.3.2. const

Fungsi const pada Arduino yaitu untuk mengubah perilaku variabel menjadi read only dengan nilai yang konstan. Sebagai contoh, penggunaan `const` biasanya digunakan untuk mendeklarasikan nomor pin perangkat. Hal tersebut dapat ditulis dengan format sebagai berikut :

```
const int ledPin = 12;
```

Keterangan:

- ledPin = variabel yang digunakan.
- 12 = nilai integer sebagai inisialisasi pin yang sedang dipakai, berada pada GPIO 12.

Sepintas penggunaan const mirip dengan #define. Perbedaan utamanya adalah bahwa penggunaan variabel dengan const akan mengikuti aturan variable scoping pada bahasa C, jadi penggunaan variabel tersebut dapat dibatasi. Sedangkan #define tidak terbatas oleh scope, jadi berlaku di semua bagian kode.

1.4. Operator

1.4.1. Operator Aritmetika

Operator	Simbol	Contoh Penggunaan

Penjumlahan	+	1 + 1 2 + 4
Pengurangan	–	1 – 1; 2 – 4;
Perkalian	*	1 * 1; 2 * 4;
Pembagian	/	1 / 1 2 / 4
Modulus	%	1 % 1 2 % 4

1.4.2. Operator Perbandingan

Operator	Simbol	Contoh Penggunaan
Lebih besar dari	>	5 > 2 (keluaran true) 4 > 5 (keluaran false)
Lebih kecil dari	<	1 < 2 (keluaran true) 4 < 2 (keluaran false)
Lebih besar sama dengan	>=	1 >= 1 (keluaran true) 2 >= 4 (keluran true)
Lebih kecil sama dengan	<=	1 <= 1 (keluaran true) 2 <= 4 (keluaran true)
Tidak sama dengan	!=	1 != 1 (keluaran false) 2 != 4 (keluaran true)
Sama dengan	==	1 == 1 (keluaran true) 2 == 4 (keluaran false)

1.4.3. Operator Logika

Operator	Penjelasan	Simbol	Contoh Penggunaan
AND	Operator and akan mengembalikan nilai true atau 1 jika kedua operand berrnilai true. Akan mengembalikan nilai false atau 0 jika salah satu atau kedua operand bernilai false	&&	`int a = 1; int b = 1; c = a && b == 1;`
OR	Operator and akan mengembalikan nilai true atau 1 jika salah satu operand berrnilai true.	||	`int a = 1; int b = 1; c = a || b == 1;`
Not	Mengembalikan nilai secara terbalik, misalnya output adalah true maka akan menjadi false	!	`int a = 1; int b = 1; c = !(a && b) == 1;`

1.5. Control Structure

Control Structure adalah struktur pengambilan keputusan untuk mengendalikan aliran pelaksanaan program. Hal ini mengharuskan programmer untuk dapat berpikir kritis ketika menyelesaikan suatu masalah tertentu melalui control statement. Dalam control statement, jika kondisi bernilai *benar* maka hasilnya *true*, sedangkan jika kondisi bernilai *salah* maka hasilnya *false*. Berikut ini adalah jenis-jenis control statement yang umum digunakan dalam pemrograman.

1.5.1. Kondisi Percabangan

1.5.1.1. Pernyataan if

Dalam Pernyataan **if**, dibutuhkan kondisi dalam tanda kurung dan pernyataan atau blok pernyataan. Jika kondisi itu benar maka pernyataan atau blok pernyataan dieksekusi kalau tidak, pernyataan ini dilewati.
Bentuk Sintaks pernyataan IF ada dua jenis, yaitu :

Bentuk 1

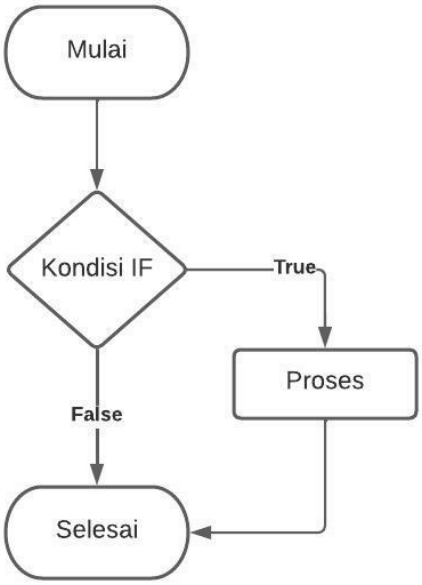
```
if (kondisi)
  pernyataan;
```

Tanda kurung { } tidak digunakan jika hanya satu pernyataan.

Bentuk 2

```
if (kondisi) {
  Blok pernyataan;
}
```

Kondisi yang dimaksud pada umumnya berbentuk operasi perbandingan, misalnya komparasi value antara variabel a dengan variabel b. Lalu blok pernyataan yang dimaksud itu meliputi semua kode yang berada di antara tanda kurung kurawal "{" dan "}". Cara kerja dari pernyataan if dapat anda lihat pada gambar 4 berikut.



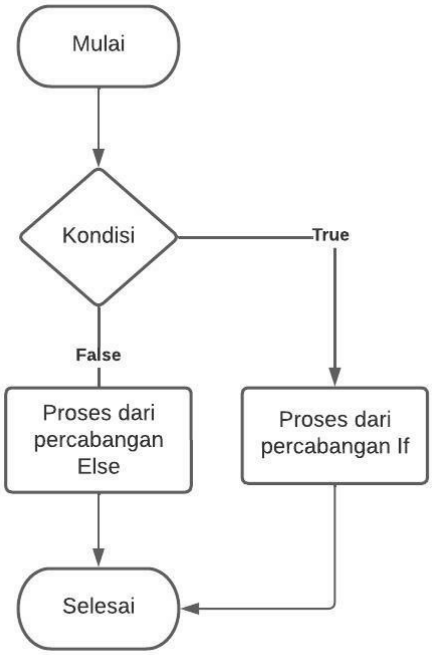
Gambar 4. Cara Kerja Pernyataan IF

1.5.1.2. Pernyataan if…else

Pernyataan **if** dapat diikuti oleh pernyataan opsional lain (**else**), yang dieksekusi ketika kondisi bernilai salah (**false**). Sintaks pernyataan **if … else** adalah sebagai berikut.

```
if (kondisi) {
  Blok pernyataan;
}
else {
  Blok pernyataan;
}
```

Cara kerja dari if else adalah ketika program memiliki kondisi yang dimana ketika program memenuhi kondisi tersebut atau true maka program akan dieksekusi. Jika program tidak memenuhi kondisi atau bernilai false maka program akan dijalankan di kondisi percabangan lainnya yaitu else. Cara kerja dari pernyataan if else dapat anda ketahui pada gambar 5 berikut.



Gambar 5. Cara Kerja Pernyataan IF - Else

1.5.1.3. Pernyataan if...else if

If else if adalah program percabangan yang terdiri dari dua atau lebih cabang yang memiliki syarat dan kondisinya masing-masing, yang dijalankan atau dieksekusi jika memenuhi kondisi yang diberikan. Jika program tidak memenuhi kondisi pertama, maka akan dialihkan ke kondisi yang kedua yaitu else if. Dan seterusnya menyesuaikan kondisi yang dibuat.

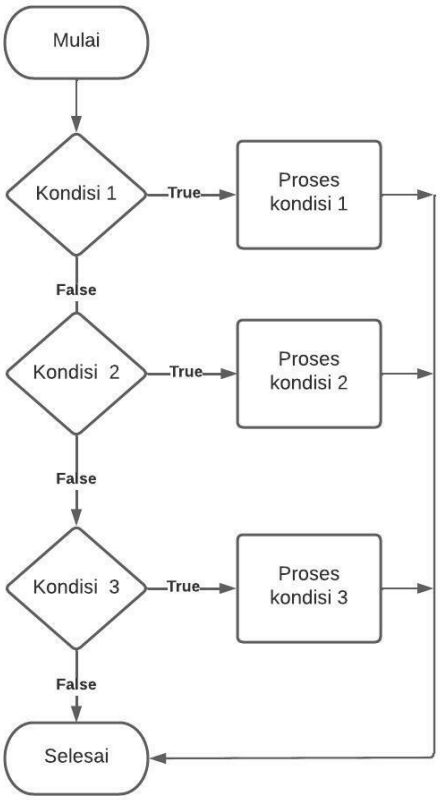
Saat menggunakan pernyataan if ... else if, perlu diingat hal-hal sebagai berikut.

- Suatu if tidak diwajibkan memiliki kondisi else.
- Suatu else if penyusunannya diwajibkan setelah if.
- Suatu if dapat memiliki jumlah else if yang banyak dengan penyusunannya diwajibkan sebelum else.
- Setelah else if sukses, maka tidak ada lagi sisa pernyataan else if atau else yang akan diuji.

Sintaks Pernyataan if ... else if

```
if (pernyataan_1) {
  Blok pernyataan;
}
else if(ekspresi_2) {
  Blok pernyataan;
}
.
.
else {
  Blok pernyataan;
}
```

Cara kerja dari pernyataan if … else if dapat anda ketahui pada gambar 6 berikut.



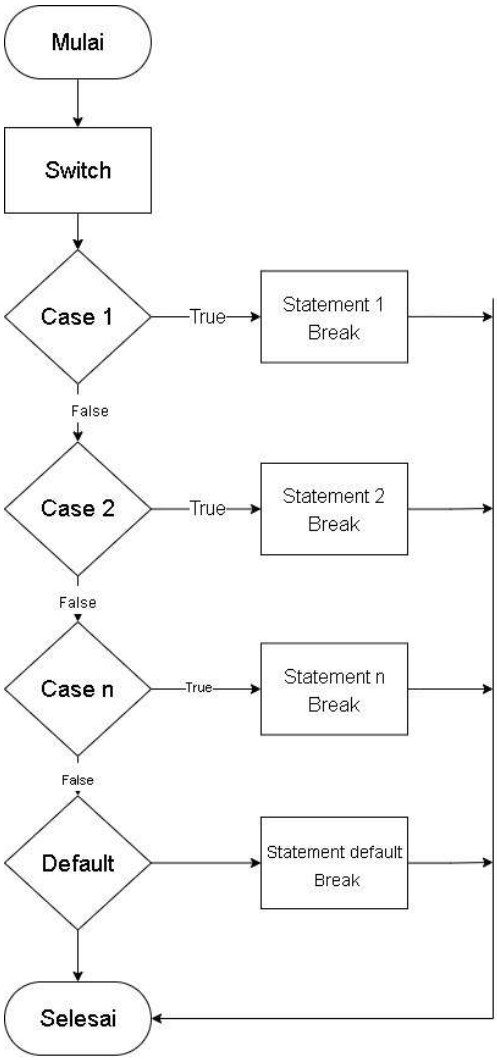
Gambar 6. Cara Kerja Pernyataan IF - Else IF

1.5.1.4. Pernyataan switch

Percabangan switch mirip dengan percabangan if else if. Pada switch terdapat case yang memiliki nilai yang berbeda-beda. Di dalam setiap case terdapat break untuk menghentikan program ketika kondisi memenuhi salah satu case. Adapun sintaks pernyataan switch dapat anda tuliskan seperti berikut.

```
switch(variabel){
    case <value>:
        // blok kode
        break;
    case <value>:
        // blok kode
        break;
    default:
        // blok kode
}
```

Cara kerja dari switch adalah ketika program memenuhi kondisi pertama atau true maka program akan dieksekusi. Jika program tidak memenuhi kondisi pertama tetapi memenuhi kondisi kedua maka program akan dijalankan di kondisi case kedua dan seterusnya. Cara kerja dari pernyataan switch dapat anda lihat pada gambar 7 berikut.



Gambar 7. Cara Kerja Pernyataan Switch

1.5.2. Kondisi Perulangan

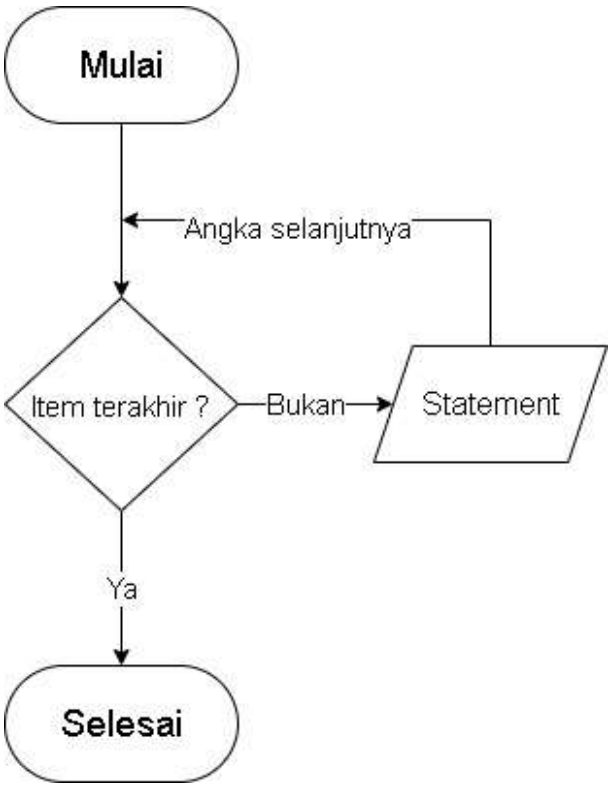
1.5.2.1. Perulangan For

Struktur perulangan for dalam aktivitas pemrograman biasanya ditulis sebagai berikut:

```
for(int i = 0; i < 10; i++){
    //Diisi dengan program yang ingin diulang;
}
```

Penjelasan struktur :

- `int i = 0` menentukan awal mulai perulangan, yaitu dimulai dari nol.
- `i < 10` adalah batas perulangan berjalan, program akan terus berulang hingga mencapai batasnya.
- `i++` , nilai pada variabel `i` ditambah satu angka setiap perulangannya.



Gambar 8. Cara Kerja Perulangan For

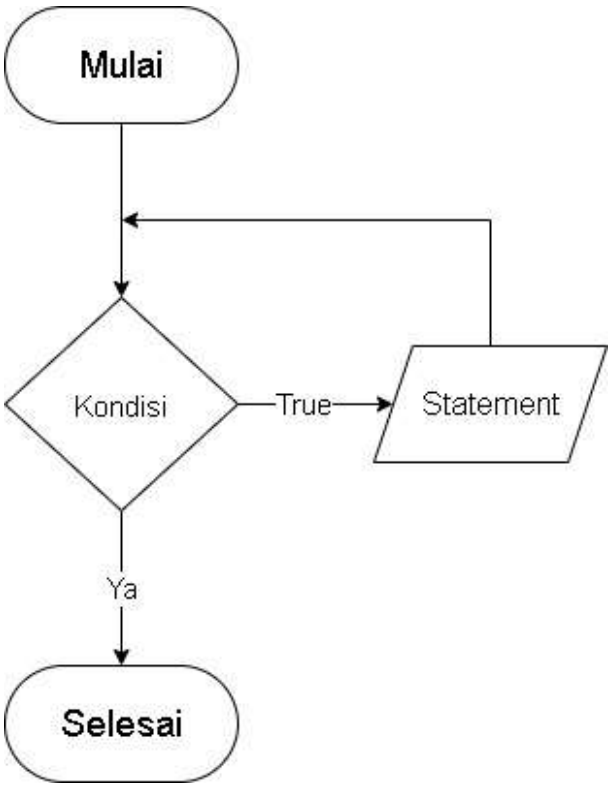
Cara kerja dari perulangan for dapat dilihat dari gambar 8 diatas, dalam contoh program sebelumnya kita mendeklarasikan awal perulangan dimulai dari 0, dan batasnya sampai angka terakhir sebelum 10. Ketika program

dijalankan, program akan melakukan perulangan hingga batas yang ditentukan, jika sudah sampai batasnya maka program akan berhenti. Berarti program perulangan akan terus berjalan selama syaratnya masih memenuhi kondisinya.

1.5.2.2. Perulangan While

While adalah perulangan yang akan terus mengeksekusi program selama memenuhi kondisi atau bernilai true yang diberikan. Contoh penulisan program :

```
int i = 0;
while (i < 5){
}
```



Gambar 9. Cara Kerja Perulangan While

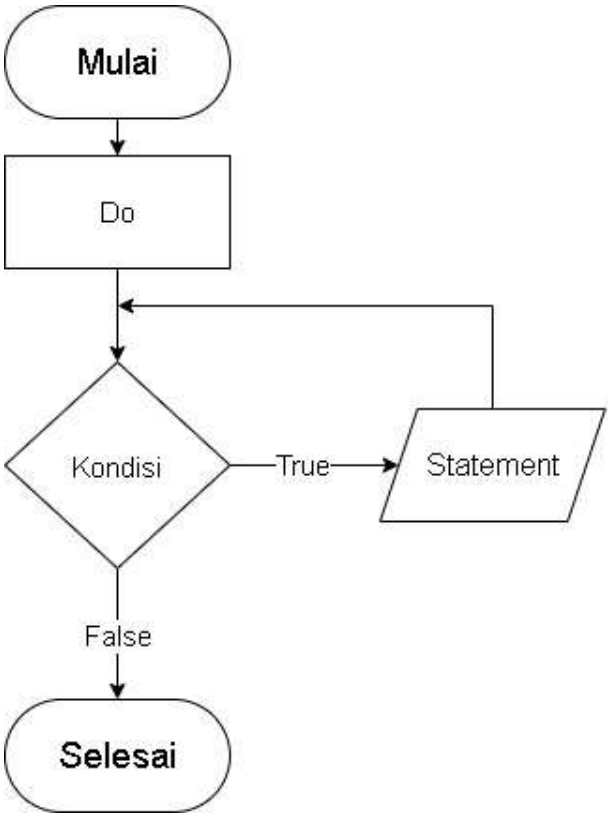
Cara kerja dari perulangan while dapat anda ketahui pada gambar 9 diatas, pada while selama kondisi masih memenuhi syarat maka program akan terus berjalan. Akan tetapi, jika kita tidak membuat batas kondisi maka perulangan akan terus berjalan dan tidak akan berhenti karena selalu memenuhi kondisi syarat.

1.5.2.3. Perulangan Do While

Mirip seperti `while` , hanya saja program dijalankan satu kali dahulu oleh `do` , lalu memeriksa kondisi oleh `while` , jika memenuhi kondisi maka perulangan akan berjalan.

Struktur do while dalam pemrograman:

```
do {
    // diisi dengan program yang ingin dijalankan
} while (<syarat kondisi>);
```



Gambar 10. Cara Kerja Perulangan Do While

Cara kerja dari perulangan do while dapat anda ketahui pada gambar 10 diatas, pada do while, program `do` akan menjalankan sekali terlebih dahulu, setelah itu `while` akan mengecek kondisi syarat. Jika memenuhi kondisi, perulangan akan berjalan dan jika tidak, maka perulangan tidak akan berjalan, hanya mencetak sekali oleh `do` .

1.5.3. Fungsi

Fungsi merupakan sub program yang membungkus program didalamnya. Fungsi dapat digunakan untuk memanggil program didalamnya.

1.5.3.1. Fungsi Tanpa Parameter

Fungsi dibuat dengan syntax `void` di awalnya. Cara menulis fungsi dalam pemrograman adalah seperti berikut.

Program :

```
void halodunia(){
    //diisi program
}

void loop(){
    // memanggil fungsi halodunia()
    halodunia();
}
```

Hasil program :

```
Halo selamat datang
```

Pada program di atas kita membuat fungsi dengan nama `halodunia` . Cara memanggil program didalam fungsi adalah dengan cara memanggil nama fungsinya yaitu `halodunia()` .

1.5.3.2. Fungsi Dengan Parameter

Program :

```
void penjumlahan(int a, int b){
  return a+b;
}

void main(){
  penjumlahan(1, 2);
  penjumlahan(3, 4);
  penjumlahan(5, 6);
}
```

Hasil program :

```
3
7
11
```

Pada program diatas terdapat dua variabel yaitu `a` dan `b` didalam fungsi. Didalam fungsi kita membuat penjumlahan dari kedua variabel dan mengembalikannya dengan `return` .

1.5.4. Array

Array merupakan tipe data terstruktur yang dapat menyimpan banyak data dengan suatu nama yang sama dan menempati tempat di memori yang berurutan serta bertipe data sama pula.

Array dalam bahasa C selalu dimulai dari indeks 0. Array dapat didefinisikan secara **statik** atau **dinamik**. Jika didefinisikan secara statik, ukuran array akan tetap dari awal program hingga akhir program. Jika didefinisikan secara dinamik, ukuran array dapat berubah selama program berjalan karena memesan tempat pada memori heap. Proses pemesanan tempat pada memori disebut dengan alokasi. Sedangkan proses pembebasan memori yang sudah dipesan disebut dengan dealokasi.

Array dapat ditulis pada pemrograman dengan format seperti berikut :

```
char variabel[panjang array] = {diisi dengan nilai}
Contoh : char alphabet[5] = {"a","b","c","d","e"}
```

Pada program di atas terdapat panjang array, panjang array adalah ukuran seberapa banyak array bisa menampung data misalnya pada contoh panjang array adalah 5 yang berarti array bisa menampung 5 data di dalamnya.

2. Alat/Instrumen/Aparatus

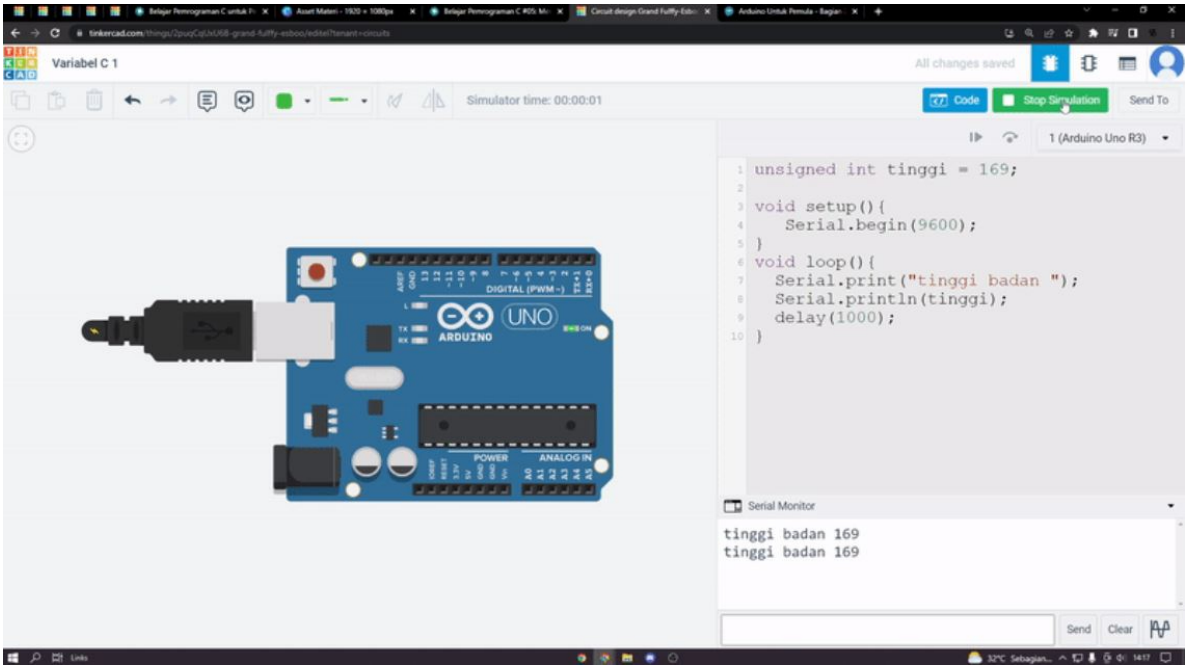
- Laptop/PC
- Platform TinkerCAD

3. Langkah Praktikum 1 – Variabel dan Tipe Data

Buatlah kode program seperti dibawah ini di Tinkercad.

```
unsigned int tinggi = 169;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("tinggi badan ");
  Serial.println(tinggi);
  delay(1000);
}
```

Setelah itu jalankan programnya.

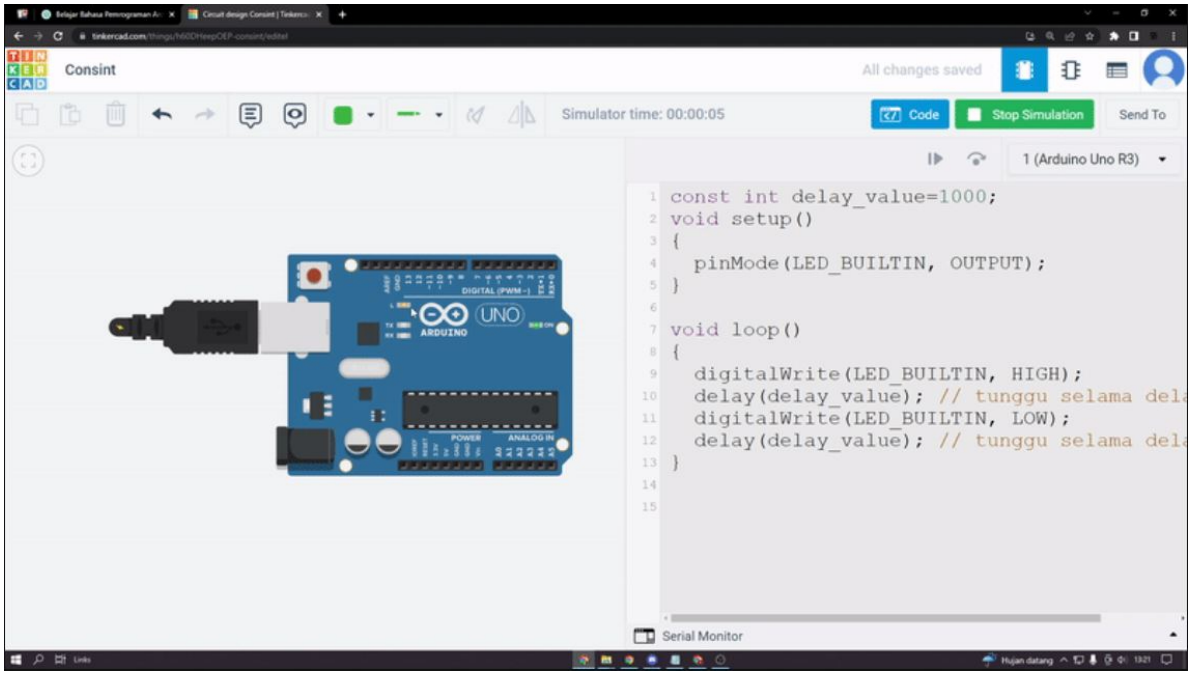


Penjelasan Praktikum :

- Pertama masuk ke Thinkercad.
- Tambahkan Arduino Uno dan masukan program seperti contoh diatas.
- Bagian `unsigned int tinggi = 169` merupakan data tinggi badan.
- `Serial.println(tinggi)` merupakan code untuk menampilkan data dari tinggi badan ke serial monitor.

4. Langkah Praktikum 2 – Konstanta

Berikut ini adalah contoh penggunaan const. Variabel delay_value dipakai untuk menentukan jangka waktu delay.

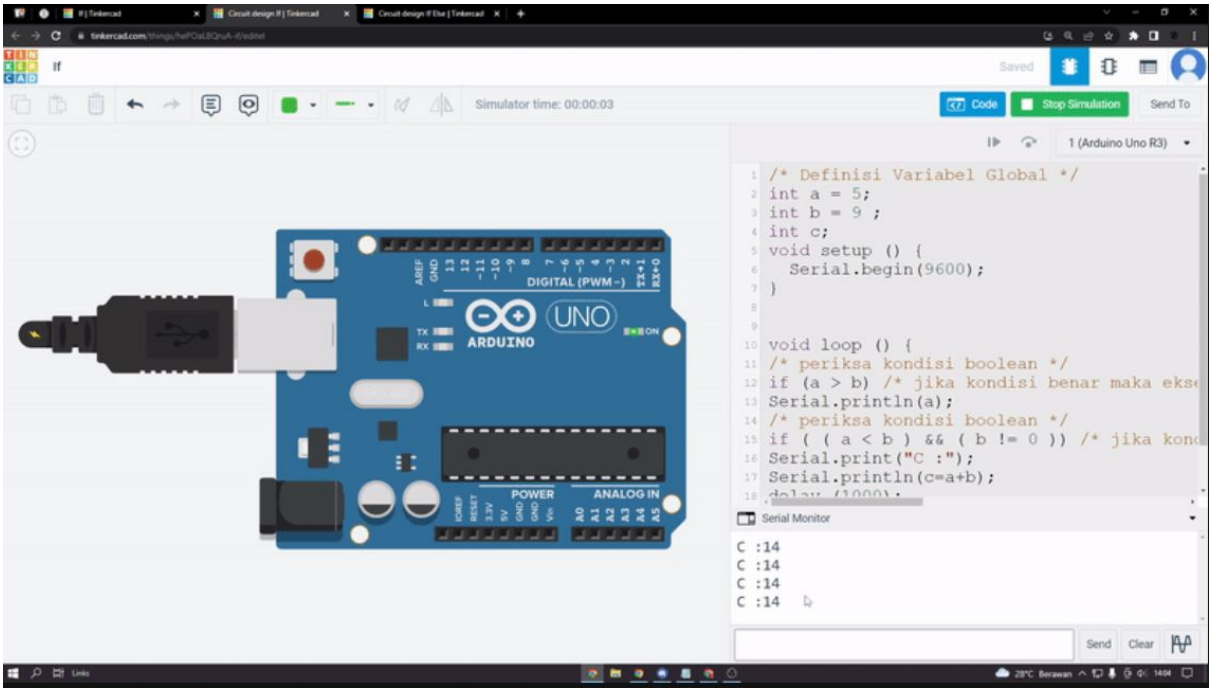


```
const int delay_value=1000;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(delay_value); // wait for a second
  digitalWrite(LED_BUILTIN, LOW);
  delay(delay_value); // wait for a second
}
```

5. Langkah Praktikum 4 – Control Structure

5.1. Praktikum menggunakan If

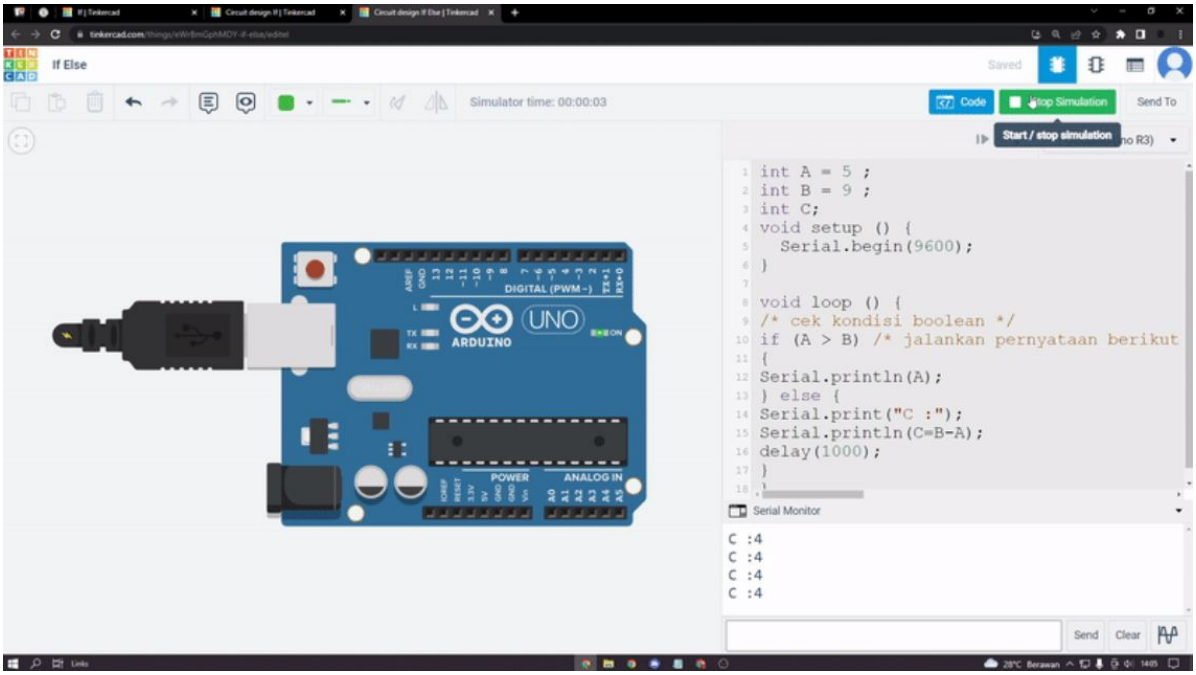
Masukkan kode program dibawah, dan jalankan di Tinkercad.



```
/* Definisi Variabel Global */
int a = 5;
int b = 9;
int c;
void setup () {
  Serial.begin(9600);
}


void loop () {
  /* periksa kondisi boolean */
  if (a > b) {/* jika kondisi benar maka eksekusi
pernyataan berikut ini, output berupa "5" */
    Serial.println(a);
  }

  /* periksa kondisi boolean */
  if ( ( a < b ) && ( b != 0 )) { /* jika kondisi
benar maka eksekusi pernyataan berikut ini, output
berupa "C:14" */
    Serial.print("C :");
    Serial.println(c=a+b);
    delay (1000);
  }
}
```

5.2 Pernyataan if...else

Masukkan kode program dibawah, dan jalankan di Tinkercad.




```
int A = 5;
int B = 9;
int C;
void setup () {
  Serial.begin(9600);
}

void loop () {
  /* cek kondisi boolean */
  if (A > B) { /* jalankan pernyataan berikut ini,
jika kondisi benar maka output yang akan keluar
tertuliskan 5, jika salah maka output yang akan keluar
tertuliskan "C:4",*/
    Serial.println(A);
  }
  else {
    Serial.print("C :");
    Serial.println(C=B-A);
    delay(1000);
  }
}
```

6. Challenge Praktikum

- Buatlah proyek dengan Thinkercad.
- Menggunakan 2 LED dan 2 Push Button.
- Gunakan fungsi “Control Structure (If Else)” di dalamnya.
- Sehingga jika tombol 1 ditekan, LED 1 Hidup dan LED 2 Mati.
- Dan Jika Tombol 2 ditekan Kedua Led akan Menyala.
- Menambahkan data output (Led On) ke Serial Monitor.