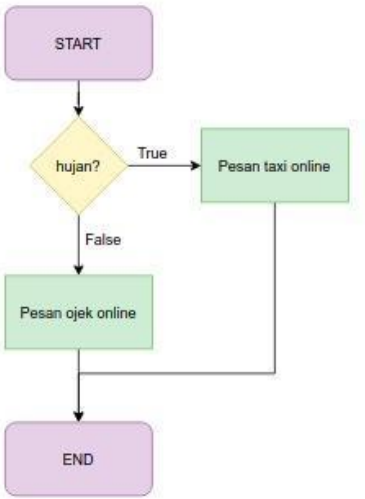
JOBSHEET 3
CONTROL FLOW PADA DART

Capaian :

- Mahasiswa mampu menerapkan percabangan if-else dalam Dart.
- Mahasiswa mampu menggunakan perulangan for dan while dalam Dart.
- Mahasiswa dapat memahami penggunaan break dan continue dalam perulangan.

Setiap hari kita melakukan perhitungan dan perbandingan guna membuat keputusan, apapun itu. Contohnya apakah perlu mencuci kendaraan ketika cuaca sedang cerah? Apa saja transportasi online yang bisa dipesan ketika hujan untuk sampai di tempat tujuan?



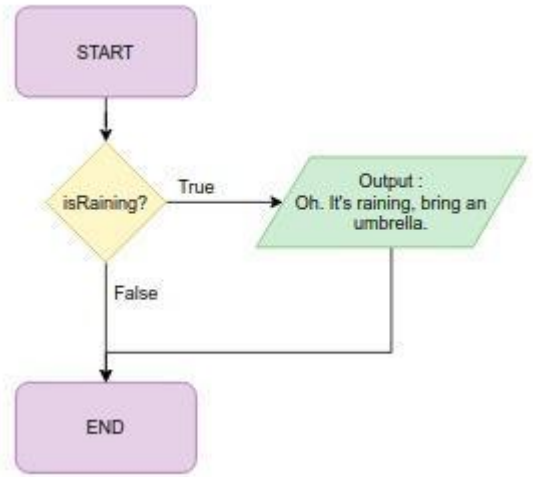
Sebuah program juga perlu membuat keputusan. Pada modul ini kita akan belajar bagaimana memberikan instruksi bagi komputer untuk mengambil keputusan dari kondisi yang diberikan serta bagaimana melakukan instruksi yang berulang.

3.1. If dan Else

Ketika mengembangkan sebuah program, kita akan bertemu dengan alur yang bercabang tergantung kepada kondisi yang terjadi. Untuk mengakomodasi dan mengecek sebuah kondisi pada Dart kita menggunakan kata kunci if.

```
void main() {
  var isRaining = true;

  print('Prepare before going to office.');
  if (isRaining) {
    print("Oh. It's raining, bring an umbrella.");
  }
  print('Going to the office.');
}
```



Kode di atas akan menampilkan output:
Prepare before going to office.
Oh. It's raining, bring an umbrella.
Going to the office.

Jika Anda mengubah nilai isRaining menjadi false, maka kode di dalam blok if akan dilewatkan dan program Anda tidak akan mengingatkan untuk membawa payung.

Lalu bagaimana jika Anda ingin melakukan operasi lain ketika kondisi bernilai false? Jawabannya adalah dengan menggunakan else. Pada contoh kode berikut kita akan melakukan pengecekan kondisi pada operator perbandingan dan operator logika.

```
void main() {
  var openHours = 8;
  var closedHours = 21;
  var now = 17;

  if (now > openHours && now < closedHours) {
```

```
    print("Hello, we're open");
  } else {
    print("Sorry, we've closed");
  }
}
```

Anda juga dapat mengecek beberapa kondisi sekaligus dengan menggabungkan else dan if. Contohnya seperti program konversi nilai berikut:

```
void main() {
  var score = 85;

  print('Nilai Anda: ${calculateScore(score)}');
}

String calculateScore(num score) {
  if (score > 90) {
    return 'A';
  } else if (score > 80) {
    return 'B';
  } else if (score > 70) {
    return 'C';
  } else if (score > 60) {
    return 'D';
  } else {
    return 'E';
  }
}
```

Fitur menarik lain dari Dart adalah conditional expressions. Dengan ini kita bisa menuliskan if-else statement hanya dalam satu baris:

```
// condition ? true expression : false expression

var shopStatus = now > openHours ? "Hello, we're open" : "Sorry, we've closed";
```

Selain itu Dart juga mendukung conditional expressions seperti berikut:

```
expression1 ?? expression2
var buyer = name ?? 'user';
```

Tugas 1

Berbeda dengan modul-modul sebelumnya yang menggunakan flowchart, bisakah kali ini Anda membuat program atau memahami maksud gambar berikut?


```
int mainCode() {
  // declaration variable
  int oilThatShouldBuy;

  // ---[ Tulis kodemu setelah baris ini ]---

  // ---[ Jangan menulis atau mengubah kode di bawah ini ]---
  return oilThatShouldBuy;
}
```

3.2. For Loops

Ketika menulis program komputer, akan ada situasi di mana kita perlu melakukan hal sama berkali-kali. Misalnya kita ingin menampilkan semua nama pengguna yang terdaftar di aplikasi kita,

atau sesederhana menampilkan angka 1 sampai 10. Tentunya tidak praktis jika kita menulis kode seperti berikut:

```
print(1);
print(2);
print(3);
print(4);
print(5);
print(6);
print(7);
print(8);
print(9);
print(10);
```

Bagaimana jika kita perlu menampilkan angka 1 sampai 100?

Dart memiliki banyak opsi untuk melakukan perulangan kode, salah satunya adalah for. For cocok digunakan pada kondisi perulangan di mana kita membutuhkan variabel indeks dan tahu berapa kali perulangan yang kita butuhkan. Sebagai contoh jika kita ingin menampilkan angka 1 sampai 100, kita bisa menuliskan seperti berikut:

```
void main() {
  for (int i = 1; i <= 100; i++) {
    print(i);
  }
}
```

Lebih ringkas bukan? Terdapat tiga bagian utama dalam sintaks for di atas:

- Pertama, variabel index yang seringkali diberi nama i yang berarti index. Pada variabel ini kita menginisialisasi nilai awal dari perulangan yang kita lakukan.
- Kedua, operasi perbandingan. Pada bagian ini komputer akan melakukan pengecekan kondisi apakah perulangan masih perlu dilakukan. Jika bernilai true maka kode di dalam blok for akan dijalankan.
- Ketiga, increment/decrement. Di sini kita melakukan penambahan atau pengurangan variabel index. Jadi pada contoh di atas variabel indeks akan ditambah dengan 1 di setiap akhir perulangan.

Jika dituliskan dalam bentuk pseudocode, maka kode di atas bisa dimaknai dengan "Jika i kurang dari sama dengan 100, maka jalankan kode berikut."

Tugas 2

Kini saatnya menguji pemahaman Anda tentang materi for loops. Bisakah Anda membuat program Dart yang menampilkan output seperti berikut?

```
*
**
***
****
*****
******
*******
********
*********
**********
```

3.3. While and do-while

do-while = akan di eksekusi 1 kali, meskipun syaratnya tidak sesuai
while-do = jika tidak sesuai persyaratan maka tidak akan di eksekusi sama sekalai

Metode lain untuk melakukan perulangan adalah dengan while. Sama seperti for, instruksi while mengevaluasi ekspresi boolean dan menjalankan kode di dalam blok while ketika bernilai true.

Untuk menampilkan angka 1 sampai 100 dengan while kita bisa menulis kode seperti berikut:

```
void main() {
  var i = 1;

  while (i <= 100) {
    print(i);
    i++;
  }
}
```

Bisa dilihat pada kode di atas bahwa perulangan dengan while tidak memiliki ketergantungan dengan variabel index seperti pada for loop. Karena itu, meskipun while dapat melakukan perulangan yang sama dengan for, while lebih cocok digunakan pada kasus di mana kita tidak tahu pasti berapa banyak perulangan yang diperlukan.

Bentuk lain dari while adalah perulangan do-while.

```
void main() {
  var i = 1;

  do {
    print(i);
    i++;
  } while (i <= 100);
}
```

Kondisi pada while akan dievaluasi sebelum blok kode di dalamnya dijalankan, sedangkan do-while akan mengevaluasi boolean expression setelah blok kodenya dijalankan. Ini artinya kode di dalam do-while akan dijalankan setidaknya satu kali.

Infinite loops

Ketika menerapkan perulangan pada program kita, ada satu kondisi yang perlu kita hindari yaitu infinite loop. Infinite loop atau endless loop adalah kondisi di mana program kita stucked di dalam perulangan. Ia akan berjalan terus hingga menyebabkan crash pada aplikasi dan komputer kecuali ada intervensi secara eksternal, seperti mematikan aplikasi.

Kode berikut ini adalah contoh di mana kondisi infinite loop dapat terjadi:

```
void main() {
  var i = 1;

  while (i < 5) {
    print(i);
  }
}
```

Dapatkah Anda mencari apa yang salah dari dari kode di atas sehingga terjadi infinite loop?

Jawabannya adalah karena variabel i selalu bernilai 1. Alhasil kondisi i < 5 akan selalu bernilai true dan akibatnya aplikasi akan terus mencetak 1 ke konsol sehingga mengalami crash.

Tugas 3

Melihat tugas sebelumnya, bisakah Anda membuat program Dart yang menampilkan output seperti di bawah ini menggunakan perulangan while atau do-while?

```
**********
*********
********
*******
******
*****
****
***
**
*
```

3.4. Break and Continue

Anda memiliki aplikasi yang menyimpan data 20 bilangan prima pertama. Pengguna dapat mencari bilangan prima lalu komputer akan menampilkan urutan berapa bilangan tersebut. Ketika bilangan prima tersebut sudah ditemukan tentunya komputer tidak perlu melanjutkan proses perulangan lagi. Nah, di sinilah kita bisa menggunakan break untuk menghentikan dan keluar dari proses iterasi.

```
void main() {
  // bilangan prima di bawah 100
  var primeNumbers = [
     2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 73, 79, 83, 89, 97
  ];

  var searchNumber = 13;
```

```
          print('Masukkan bilangan prima di antara 1-100: $searchNumber');

      for (int i = 0; i < primeNumbers.length; i++) {
        if (searchNumber == primeNumbers[i]) {
          print('$searchNumber adalah bilangan prima ke-${i + 1}');
          break;
        }
        print('$searchNumber != ${primeNumbers[i]}');
      }
    }

    /// Output:
    /// Masukkan bilangan prima : 13
    /// 13 != 2
    /// 13 != 3
    /// 13 != 5
    /// 13 != 7
    /// 13 != 11
    /// 13 adalah bilangan prima ke-6
```

Ketika kode di atas dijalankan, proses iterasi akan dihentikan ketika angka yang diinputkan pengguna sama dengan bilangan prima yang dicari.

```
Masukkan bilangan prima : 13
13 != 2
13 != 3
13 != 5
13 != 7
13 != 11
13 adalah bilangan prima ke-6
```

Keyword lain yang berguna pada proses perulangan adalah continue. Dengan continue kita bisa melewatkan proses iterasi dan lanjut ke proses iterasi berikutnya. Misalnya Anda ingin menampilkan angka 1 sampai 10 kecuali angka kelipatan 3. Anda dapat menuliskannya seperti berikut:

```
void main() {
  for (int i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
      continue;
    }
    print(i);
  }
}

/// Output:
/// 1
/// 2
/// 4
/// 5
/// 7
/// 8
/// 10
```

3.5. Switch and Case

Sebelumnya kita telah mempelajari bagaimana mengondisikan logika komputer dengan menggunakan if. Namun, bagaimana jika ada banyak kondisi yang perlu dicek menggunakan if? Tentu akan membingungkan dan kode kita pun jadi sulit dibaca.

Dart mendukung statement switch untuk melakukan pengecekan banyak kondisi dengan lebih mudah dan ringkas.

```
switch (variable/expression) {
  case value1:
    // do something
    break;
  case value2:
    // do something
    break;
  ...
  ...
  default:
    // do something else
}
```

Tanda kurung setelah keyword switch berisi variabel atau ekspresi yang akan dievaluasi. Kemudian untuk setiap kondisi yang mungkin terjadi kita masukkan keyword case diikuti dengan nilai yang valid. Jika kondisi pada case sama dengan variabel pada switch, maka blok kode setelah titik dua (:) akan dijalankan. Keyword break digunakan untuk keluar dari proses switch. Terdapat satu case bernama default yang memiliki fungsi yang sama dengan keyword else pada control flow if-else. Jika tidak ada nilai yang sama dengan variabel pada switch maka blok kode ini akan dijalankan.

Berikut ini adalah contoh aplikasi kalkulator yang menerapkan switch-case.

```
void main() {
  final firstNumber = 13;
  final secondNumber = 18;
  final operator = "+";

  switch (operator) {
    case '+':
```

```
        print(
                        '$firstNumber $operator $secondNumber =
${firstNumber + secondNumber}');
        break;
      case '-':
        print(
                        '$firstNumber $operator $secondNumber =
${firstNumber - secondNumber}');
        break;
      case '*':
        print(
                        '$firstNumber $operator $secondNumber =
${firstNumber * secondNumber}');
        break;
      case '/':
        print(
                        '$firstNumber $operator $secondNumber =
${firstNumber / secondNumber}');
        break;
      default:
        print('Operator tidak ditemukan');
    }
  }
```

Tugas 4

Jadikan contoh program diatas bisa menerima masukkan dari user.

Contoh:

Input: Masukkan bilangan 1: 16 Masukkan bilangan 2: 4 Masukkan operator: / Output: Hasilnya dari 16 / 4 adalah 4

Input: Masukkan bilangan 1: 16
Masukkan bilangan 2: 4
Masukkan operator: /
Output: Hasilnya dari 16 / 4 adalah 4