MODUL 2
BAHASA PEMOGRAMAN DART (2)

Control Flow Statements

Dart menyediakan beberapa perintah yang dapat digunakan untuk mengontrol aliran kode.

- if and else
- switch and case
- for loops
- while and do-while loops
- break and continue
- assert

If and Else

Dalam pernyataan ini hanya memeriksa kondisi dan jika benar, pernyataan di dalamnya akan dieksekusi tetapi jika tidak maka pernyataan lain akan dieksekusi.

```dart
main() {
  var pointsA = 50;
  var pointsB = 50;

  if (pointsA > pointsB) {
    print("Tim A menang!");
  } else if (pointsB > pointsA) {
    print("Tim B menang!");
  } else {
    print("Seri!");
  }
}
```

Untuk menyederhanakan if else, kita dapat menggunakan operator **ternary**. Struktur kondisi menggunakan operator ternary adalah sebagai berikut:

```
condition ? expression1 : expression2
```

```dart
main() {
  var a = 5;
  var b = 2;
  var result;

  if (a > b) {
    result = a - b;
  } else {
    result = b - a;
  }
  print(result);

  // kondisi diatas dapat diganti dengan operator ternary berikut ini
  var result2 = a > b ? a - b : b - a;
  print(result2);
}
```

Switch and Case

Switch case mirip dengan if else namun lebih terstruktur dan mudah dipahami.

```
main() {
  var nilai = 75;

  if (nilai >= 85) {
    print("A");
  } else if (nilai >= 75) {
    print("B");
  } else if (nilai >= 65) {
    print("C");
  } else if (nilai > 0) {
    print("D");
  } else {
    print("E");
  }

  switch (nilai) {
    case >= 85:
      print("A");
      break;
    case >= 75:
      print("B");
      break;
    case >= 65:
      print("C");
      break;
    case > 0:
      print("D");
      break;
    default:
      print("E");
  }
}

```

For

For digunakan untuk melakukan perulangan dengan jumlah yang sudah diketahui diawal.

```
main() {
  var colorList = ['blue', 'yellow', 'green', 'red'];
  for (var i = 0; i < colorList.length; i++) {
    print(colorList[i]);
  }
  print("-----");

  // alternatif
  for (var i in colorList) {
    print(i);
  }
}
```

While, Do While

While dan do while digunakan jika kita belum mengetahui jumlah perulangan yang harus dijalankan.

```
main() {
  // while
  var i = 0;
  while (i <= 10) {
    print(i);
    i++;
  }

  print("-----");

  // do while
  i = 0;
  do {
    print(i);
    i++;
  } while (i <= 10);
}
```

Apa perbedaan while dan do while?

While Challenge!

Anda memiliki sebuah oven yang memiliki suhu awal adalah 50, dan anda perlu memanaskannya hingga 250. Jika suhu oven belum mencapai 250, maka naikkan suhu sebesar 10. Berapa kali oven harus dinaikkan suhunya hingga mencapai suhu 250?

Lengkapi kode berikut ini untuk menyelesaikan masalah diatas.

```
main() {
  var temperatur = 50;
  var i = 0;

  // tuliskan kode anda disini

  print("Temperatur dinaikan sebanyak $i kali");
}
```

Break and Continue

Break digunakan untuk menghentikan perulangan, sedangkan *continue* digunakan untuk melanjutkan perulangan walaupun masih terdapat kode dibawahnya.

```
main() {
  var intList = [7, 3, 9, 6, 2, 5, 4];

  for (var element in intList) {
    if (element % 2 == 0) {
      print(element);
      break;
    }
  }

  print("-----");

  var nilai = [8, 7, 4, 3, 5, 9, 6];

  for (var element in nilai) {
    if (element < 6) {
      continue;
    }
    print(element);
  }
}

```

Assert

Assert adalah statement yang sangat berguna yang memungkinkan kita memberikan kondisi pada eksekusi kode. Ini digunakan untuk menghentikan eksekusi normal ketika kondisi boolean salah. Biasanya assert digunakan untuk melakukan debuging.

```
main() {
  var nilai = 75.8;
  print(nilai);

  assert(nilai.runtimeType == int);

  print("kode berikutnya");
}
```

Function

Struktur fungsi adalah sebagai berikut :

```
// Tanpa parameter
returnType functionName(){
  function body
}

//dengan parameter
returnType functionName(parameters){
  function body
}

// fungsi dengan satu ekspresi
returnType functionaName(parameter)=> function body
```

```dart
num sum(num x, num y) {
  return x + y;
}

num sum2(num x, num y) => x + y;

main() {
  print(sum(1, 2));
  print(sum2(1, 2));
}

```

Parameters

Terdapat dua cara membuat parameter pada sebuah fungsi, parameter sesuai urutan (positional parameters) dan parameter bernama (named parameters). Untuk membuat parameter bernama, letakkan parameter didalam tanda {}. Jika kita meletakkan parameter di dalam [] maka parameter tersebut bersifat opsional. Jika kita meletakkan karakter? di belakang tipe data, artinya parameter tersebut dapat berisi null (kosong).

```dart
// Positional parameters
num subtraction(num x, num y) {
  return x - y;
}

// Named parameters
num subtraction2({required num x, required num y}) {
  return x - y;
}

// Optional parameters
num subtraction3(num x, [num y = 0]) {
  return x - y;
}

// Nullable parameters
num calc(num x, num y, num? z) {
  if (z != null) {
    return x + y + z;
  } else {
    return x + y;
  }
}
main() {
  print(subtraction(5, 2));
  print(subtraction2(x: 5, y: 2));
  print(subtraction2(y: 2, x: 5));
  print(subtraction3(5));
  print(calc(1, 2, null));
}
```

Anonymous Function

Ada kalanya kita hanya perlu menggunakan fungsi satu kali, atau sementara, hanya membutuhkan fungsionalitas dari fungsi tersebut. Kita dapat menggunakan anonymous function atau fungsi tidak bernama.

```dart
int hitung(nilai) {
  return nilai * 2;
}

main() {
  List<int> nilai = [1, 2, 3, 4, 5];

  for (int i in nilai) {
    print(hitung(i));
  }

  print("---");

  // Dengan anonymous function kita tidak perlu membuat fungsi baru
  nilai.forEach((element) {
    print(element * 2);
  });
}
```

Object Oriented

Berikut ini contoh penerapan OOP di dart.

```dart
class Person {
  // Properties
  String? name;
  String? gender;
  int? age;

  // Constructor
  Person(this.name, this.gender, [this.age]);

  // Getter
  String get hello => "Hello my name is $name";

  // Method
  walking() => print('$name is walking');
  talking() => print('$name is talking');
}

main() {
  Person person = Person("Budi", "Male");
  print(person.name);
  person.age = 25;
  print(person.age);
  print(person.hello);
  person.walking();
}
```