Dasar-Dasar Pemrograman Dart

---

1. Tipe Data, Variabel, dan Konstanta pada Dart

📌 1.1 Tipe Data di Dart

Dart adalah bahasa yang **statically typed**, artinya setiap variabel memiliki tipe data tertentu. Berikut tipe-tipe data dasar:

Tipe Data	Contoh	Keterangan
`int`	`int x = 10;`	Bilangan bulat
`double`	`double y = 3.14;`	Bilangan desimal
`String`	`String name = "Dart";`	Teks/kalimat
`bool`	`bool isActive = true;`	Boolean: `true` atau `false`
`List`	`List<int> numbers = [1, 2, 3];`	Daftar/array
`Map`	`Map<String, int> ages = {'Ali': 20};`	Key-value pair
`dynamic`	`dynamic data = 42;`	Tipe data fleksibel

📌 1.2 Variabel

Variabel di Dart bisa dideklarasikan dengan tipe eksplisit atau menggunakan kata kunci `var`, `final`, atau `const`.

```
var name = 'Ali'; // tipe otomatis: String
int age = 20;
double height = 1.75;
```

📌 1.3 Konstanta (`final` vs `const`)

- `final` → Nilai hanya diatur sekali saat runtime.
- `const` → Nilai bersifat konstan pada saat kompilasi.

```
final currentDate = DateTime.now(); // boleh berubah setiap program
dijalankan
const pi = 3.14; // nilai tetap, tidak boleh berubah
```

---

2. Struktur Kontrol Alur

📌 2.1 Percabangan: `if`, `else`, dan `else if`

```
int age = 20;
```

```
if (age >= 18) {
  print("Dewasa");
} else if (age >= 13) {
  print("Remaja");
} else {
  print("Anak-anak");
}
```

📌 2.2 `switch-case`

Digunakan saat ada banyak kondisi terhadap 1 variabel.

```
String grade = 'B';

switch (grade) {
  case 'A':
    print("Sangat baik");
    break;
  case 'B':
    print("Baik");
    break;
  default:
    print("Perlu perbaikan");
}
```

📌 2.3 Perulangan (`loop`)

`for` loop

```
for (int i = 0; i < 5; i++) {
  print(i);
}
```

`while` loop

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

`do-while` loop

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

`for-in` loop (digunakan untuk list)

```
List<String> fruits = ['apel', 'jeruk', 'pisang'];

for (var fruit in fruits) {
  print(fruit);
```

```
}
```

---

3. Fungsi dan Metode dalam Dart

📌 3.1 Fungsi Dasar

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

Pemanggilan:

```
print(tambah(3, 4)); // Output: 7
```

📌 3.2 Fungsi Tanpa Tipe Kembali (void)

```
void sapa(String nama) {
  print("Halo, $nama!");
}
```

📌 3.3 Parameter Opsional & Default

```
void showProfile(String name, [int age = 0]) {
  print("Nama: $name, Umur: $age");
}
```

📌 3.4 Fungsi sebagai Expression (Arrow Function)

```
int kali(int a, int b) => a * b;
```

📌 3.5 Fungsi Anonim dan Lambda

```
var angka = [1, 2, 3];
angka.forEach((item) {
  print(item);
});
```

---

4. Pemahaman Konsep `async` dan `await` dalam Dart

📌 4.1 Apa itu `async` dan `await`?

- `async` digunakan untuk menandai fungsi yang bersifat asynchronous (tidak dijalankan secara langsung).
- `await` digunakan untuk menunggu proses asynchronous selesai sebelum melanjutkan eksekusi.

📌 4.2 Contoh Fungsi Asynchronous

```
Future<String> fetchData() async {
  await Future.delayed(Duration(seconds: 2));
  return 'Data berhasil diambil';
}
```

📌 4.3 Pemanggilan Fungsi Async

```
void main() async {
  print("Memulai...");
  String data = await fetchData();
  print(data);
  print("Selesai");
}
```

Output:

```
Memulai...
(Data akan muncul setelah 2 detik)
Data berhasil diambil
Selesai
```

📌 4.4 Error Handling pada Async

```
void main() async {
  try {
    String data = await fetchData();
    print(data);
  } catch (e) {
    print('Terjadi kesalahan: $e');
  }
}
```

📌 Rangkuman

Konsep	Penjelasan Singkat
Tipe Data	Dart mendukung int, double, String, bool, List, Map, dan dynamic.
Variabel & Konstanta	Gunakan `var`, `final`, atau `const` untuk deklarasi.
Kontrol Alur	Gunakan `if-else`, `switch`, dan berbagai jenis loop.
Fungsi	Dapat memiliki parameter, nilai kembali, dan bersifat async.
Async/Await	Untuk menangani operasi asynchronous dengan `Future`.

Contoh program sederhana :

```
import 'dart:io';

double tambah(double a, double b) => a + b;
double kurang(double a, double b) => a - b;
double kali(double a, double b) => a * b;
double bagi(double a, double b) {
  if (b == 0) throw Exception("Tidak bisa bagi dengan nol");
  return a / b;
}

void main() {
  stdout.write("Masukkan angka pertama: ");
  double angka1 = double.parse(stdin.readLineSync()!);

  stdout.write("Masukkan angka kedua: ");
  double angka2 = double.parse(stdin.readLineSync()!);

  stdout.write("Pilih operasi (+, -, *, /): ");
```

```
  String operasi = stdin.readLineSync()!;

  double hasil;

  try {
    switch (operasi) {
      case '+':
        hasil = tambah(angka1, angka2);
        break;
      case '-':
        hasil = kurang(angka1, angka2);
        break;
      case '*':
        hasil = kali(angka1, angka2);
        break;
      case '/':
        hasil = bagi(angka1, angka2);
        break;
      default:
        print("Operasi tidak dikenali.");
        return;
    }

    print("Hasil: $hasil");

  } catch (e) {
    print("Terjadi kesalahan: $e");
  }
}
```

Langkah-Langkah Menjalankan Program Dart di VS Code

📥 1. Install Dart SDK

Jika belum menginstalnya:

Opsi A – Jika belum punya Flutter:

- Unduh Dart SDK langsung dari:
  https://dart.dev/get-dart

Opsi B – Jika sudah install Flutter :

- Dart SDK sudah otomatis terinstal bersama Flutter.

Cek dengan:

```
dart --version
```

✒️ 2. Install VS Code Extension: Dart

Buka VS Code, lalu:

- Tekan `Ctrl + Shift + X` (untuk buka Extensions)
- Cari: `Dart`
- Install ekstensi resmi dari Dart Code

Tips: Kalau kamu juga akan pakai Flutter, install juga ekstensi **Flutter**

🗀 3. Buat Folder dan File Dart

1. Buka VS Code
2. Buat folder baru, misalnya: `dart_kalkulator`
3. Buat file baru: `main.dart`
4. Salin & tempelkan kode kamu ke dalam `main.dart`

□ 4. Jalankan Program dari Terminal

Pastikan kamu membuka **terminal VS Code** di folder yang berisi file tersebut:

```
cd path/to/dart_kalkulator
dart run main.dart
```

📌 Jika kamu menyimpan file dengan nama lain, pastikan kamu jalankan sesuai namanya:

```
dart run nama_file.dart
```

🗂 5. Contoh Struktur Folder

```
dart_kalkulator/
└── main.dart
```

📝 Tips Tambahan

- `import 'dart:io';` hanya bisa dijalankan **bukan di browser**, tapi di CLI (command line interface)
- Jadi, **tidak bisa dijalankan di DartPad**, harus via terminal lokal

---

🧪 Uji Coba

Setelah mengetik perintah `dart run main.dart`, kamu akan melihat prompt seperti:

```
Masukkan angka pertama:
```

Masukkan angka, lalu ikuti petunjuk selanjutnya untuk melakukan operasi matematika.

🚀 Kalau Mau Lebih Praktis

Tambahkan juga file `pubspec.yaml` jika ingin membuat proyek Dart standar:

```
name: dart_kalkulator
description: A simple command-line calculator
environment:
  sdk: '>=2.12.0 <3.0.0'
```

Kemudian jalankan:

```
dart pub get
dart run bin/main.dart
```

Tapi untuk proyek kecil seperti ini, cukup satu file `.dart` saja juga sudah cukup.

Gunakan:

```
dart create dart_kalkulator
```

Ini membuat proyek **Dart command-line (CLI)**, cocok untuk program seperti kalkulator kamu yang menggunakan `dart:io`.

✅ Membuat proyek Flutter (UI, aplikasi mobile/web/desktop)

Gunakan:

```
flutter create dart_kalkulator
```

Ini membuat proyek **Flutter**, yang berisi UI, widget, dan file starter Flutter. Bukan untuk command-line biasa.
