**LAPORAN PRAKTIKUM**

**Jobsheet 4 : Pengantar Bahasa Pemrograman Dart - Bagian 3**

Disusun untuk memenuhi tugas mata kuliah Pemrograman Mobile

Dosen Pengampu :

Samsul Arifin, S.Kom.MMSI

****

**DISUSUN OLEH:**

**PUTRI AYU ALICIAWATI**

**2241720132**

**TI-3C**

**D-IV TEKNIK INFORMATIKA**

**TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2023/2024**

**Tugas**

1. Silakan selesaikan Praktikum 1 sampai 5, lalu dokumentasikan berupa screenshot hasil pekerjaan Anda beserta penjelasannya!
2. Jelaskan yang dimaksud Functions dalam bahasa Dart!

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi memungkinkan untuk mengorganisir kode dengan cara yang modular dan terstruktur. Fungsi dapat didefinisikan untuk melakukan operasi tertentu, menerima argumen, dan mengembalikan nilai.

1. Jelaskan jenis-jenis parameter di Functions beserta contoh sintaksnya!

Parameter fungsi dapat dikategorikan menjadi beberapa jenis berdasarkan cara mereka didefinisikan dan digunakan.

* Parameter Positional (Parameter Posisi)

Parameter yang harus disediakan dalam urutan tertentu saat memanggil fungsi. Jika urutan atau jumlah parameter tidak sesuai, maka akan terjadi kesalahan.

Contoh sintaks :

void main() {

  // Memanggil fungsi

  sapa('Putri', 21);

}

void sapa(String nama, int usia) {

  print('Halo, $nama! Usia Anda adalah $usia.');

}

* Parameter Default

Parameter yang memiliki nilai default jika tidak diberikan nilai saat pemanggilan fungsi. Parameter ini didefinisikan dengan tanda = diikuti dengan nilai default.

Contoh :

void main() {

  // Memanggil fungsi

  sapa('Putri');

}

void sapa(String nama, [int usia = 20]) {

  print('Halo, $nama! Usia Anda adalah $usia.');

}

* Parameter Named (Parameter Bernama)

Parameter yang didefinisikan dengan nama, memungkinkan pemanggilan fungsi dengan menyebutkan nama parameter secara eksplisit. Parameter ini diletakkan dalam kurung kurawal {} saat mendefinisikan fungsi.

Contoh :

void main() {

  // Memanggil fungsi dengan satu parameter bernama

  sapa(nama: 'Putri');

}

void sapa({required String nama, int usia = 18}) {

  print('Halo, $nama! Usia Anda adalah $usia.');

}

* Parameter Opsional Positional (Parameter Posisi Opsional)

Parameter yang tidak wajib disediakan saat pemanggilan fungsi, didefinisikan dalam kurung kotak []. Jika tidak diberikan, nilai default akan digunakan jika ada.

Contoh :

void main() {

  // Memanggil fungsi dengan parameter opsional

  sapa('Putri');

}

void sapa(String nama, [int usia = 1, String sapaan = 'Halo']) {

  print('$sapaan, $nama! Usia Anda adalah $usia.');

}

* Parameter Opsional Named (Parameter Bernama Opsional)

Parameter bernama yang tidak wajib disediakan, didefinisikan dalam kurung kurawal {} dan dapat memiliki nilai default.

Contoh :

void main() {

  // Memanggil fungsi dengan parameter bernama

sapa();

}

void sapa({String nama = 'Teman', int usia = 18, String sapaan = 'Halo'}) {

  print('$sapaan, $nama! Usia Anda adalah $usia.');

}

1. Jelaskan maksud Functions sebagai first-class objects beserta contoh sintaknya!

Fungsi adalah first-class objects (objek kelas pertama), yang berarti fungsi dapat diperlakukan seperti objek lainnya. Sebagai objek kelas pertama, fungsi memiliki kemampuan berikut:

1. Dapat Disimpan dalam Variabel: Fungsi dapat disimpan dalam variabel, memungkinkan Anda untuk memanggil fungsi melalui variabel tersebut.

Contoh :

void main() {

// Menyimpan fungsi dalam variabel

Function tambah = (int a, int b) => a + b;

// Memanggil fungsi melalui variabel

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

}

1. Dapat Dipassing Sebagai Argumen: Fungsi dapat diteruskan sebagai argumen ke fungsi lain.

Contoh :

void main() {

void cetakHasil(int a, int b, Function(int, int) operasi) {

print(operasi(a, b));

}

// Mendefinisikan fungsi operasi

int tambah(int a, int b) => a + b;

// Mempassing fungsi ke fungsi lain

cetakHasil(3, 4, tambah); // Output: 7

}

1. Dapat Dihasilkan oleh Fungsi Lain: Fungsi dapat dikembalikan sebagai hasil dari fungsi lain.

Contoh :

Function(int) buatPengganda(int faktor) {

return (int nilai) => nilai \* faktor;

}

void main() {

// Mendapatkan fungsi pengganda

var penggandaDua = buatPengganda(2);

// Menggunakan fungsi pengganda

print(penggandaDua(5)); // Output: 10

}

1. Dapat Disimpan dalam Koleksi: Fungsi dapat disimpan dalam list, set, atau map.

Contoh :

void main() {

// Mendefinisikan beberapa fungsi

var fungsiList = [

(int a, int b) => a + b,

(int a, int b) => a - b,

(int a, int b) => a \* b,

];

// Menggunakan fungsi dari list

print(fungsiList[0](3, 4)); // Output: 7 (Penjumlahan)

print(fungsiList[1](5, 2)); // Output: 3 (Pengurangan)

print(fungsiList[2](3, 4)); // Output: 12 (Perkalian)

}

1. Apa itu Anonymous Functions? Jelaskan dan berikan contohnya!

Anonymous functions (fungsi anonim) dalam Dart adalah fungsi yang tidak memiliki nama. Fungsi ini sering digunakan untuk operasi singkat, seperti dalam kasus callback atau operasi inline yang tidak memerlukan fungsi dengan nama terpisah. Fungsi anonim dapat didefinisikan dan digunakan secara langsung tanpa harus mendefinisikan fungsi dengan nama terlebih dahulu.

1. Fungsi Anonim sebagai Argument

Fungsi anonim sering digunakan sebagai argumen untuk fungsi lain, terutama dalam operasi seperti sorting atau filtering. Contoh :

void main() {

var angka = [3, 1, 4, 1, 5, 9];

// Menggunakan fungsi anonim untuk menyortir list

angka.sort((a, b) => a.compareTo(b));

print(angka); // Output: [1, 1, 3, 4, 5, 9]

}

1. Fungsi Anonim dalam Callback

Fungsi anonim sering digunakan dalam callback untuk menangani event atau hasil operasi. Contoh :

void main() {

void operasi(int a, int b, Function(int, int) callback) {

print(callback(a, b));

}

// Menggunakan fungsi anonim sebagai callback

operasi(5, 3, (a, b) => a + b); // Output: 8

}

1. Fungsi Anonim dalam List

Fungsi anonim juga dapat disimpan dalam list dan dipanggil sesuai kebutuhan. Contoh :

void main() {

// Menyimpan fungsi anonim dalam list

var fungsiList = [

(int a, int b) => a + b,

(int a, int b) => a - b,

(int a, int b) => a \* b,

];

// Menggunakan fungsi dari list

print(fungsiList[0](10, 5)); // Output: 15 (Penjumlahan)

print(fungsiList[1](10, 5)); // Output: 5 (Pengurangan)

print(fungsiList[2](10, 5)); // Output: 50 (Perkalian)

}

1. Jelaskan perbedaan Lexical scope dan Lexical closures! Berikan contohnya!
2. Lexical Scope

Lexical Scope mengacu pada aturan bagaimana variabel diakses berdasarkan lokasi di mana variabel tersebut dideklarasikan dalam kode. Dengan lexical scoping, scope (jangkauan) variabel ditentukan oleh struktur kode pada saat penulisan (lexical atau statik).

Contoh :

void main() {

int a = 10; // variabel a berada dalam scope global

void fungsi() {

int b = 20; // variabel b berada dalam scope fungsi

print(a); // mengakses variabel a dari scope global

print(b); // mengakses variabel b dari scope fungsi

}

fungsi();

// print(b); // Error: b tidak dikenal di scope ini

}

1. Lexical Closures

Lexical Closures mengacu pada kemampuan fungsi untuk "menutup" atau menyimpan referensi ke variabel yang ada di luar fungsi tersebut (lingkungan lexical) ketika fungsi itu diciptakan. Dengan kata lain, fungsi dapat mengakses variabel-variabel di luar scope-nya saat fungsi tersebut dibuat.

Contoh :

void main() {

Function createCounter() {

int count = 0; // variabel count berada dalam scope fungsi createCounter

// Fungsi anonim yang merupakan closure

return () {

count++;

print(count);

};

}

var counter = createCounter(); // Mengambil fungsi closure

counter(); // Output: 1

counter(); // Output: 2

}

1. Jelaskan dengan contoh cara membuat return multiple value di Functions!

Dua metode yang umum digunakan adalah dengan menggunakan Tuple (Record) atau dengan menggunakan objek custom.

1. Tuple (Record): Cocok untuk mengembalikan beberapa nilai dalam satu entitas yang sederhana, tanpa perlu mendefinisikan struktur khusus. Cocok untuk penggunaan sementara.

Contoh :

void main() {

// Fungsi yang mengembalikan Record

(int, String, bool) fungsi() {

return (42, 'Hello', true);

}

// Memanggil fungsi dan mendapatkan hasil

var hasil = fungsi();

// Mengakses nilai dari Record

print(hasil.$1); // Output: 42

print(hasil.$2); // Output: Hello

print(hasil.$3); // Output: true

}

1. Objek Custom: Lebih fleksibel dan terstruktur, memungkinkan Anda untuk memberi nama dan mendokumentasikan nilai-nilai yang dikembalikan, serta memberikan metode tambahan jika diperlukan.

Contoh :

class Hasil {

final int angka;

final String teks;

final bool flag;

Hasil(this.angka, this.teks, this.flag);

}

void main() {

// Fungsi yang mengembalikan objek Hasil

Hasil fungsi() {

return Hasil(42, 'Hello', true);

}

// Memanggil fungsi dan mendapatkan hasil

var hasil = fungsi();

// Mengakses nilai dari objek Hasil

print(hasil.angka); // Output: 42

print(hasil.teks); // Output: Hello

print(hasil.flag); // Output: true

}

1. Kumpulkan berupa link commit repo GitHub pada tautan yang telah disediakan di grup Telegram!