Les entrées et sorties en Dart

1. Objectif

- L'objectif de ce cours est de vous familiariser avec les concepts et les techniques liés aux entrées/sorties (I/O) en Dart. Vous apprendrez comment interagir avec le système de fichiers, lire depuis la console, écrire dans des fichiers, et effectuer des requêtes HTTP en utilisant les bibliothèques dart:io et http.
- À la fin de ce cours, vous serez capable de créer des programmes Dart capables de manipuler des données à partir de sources externes et de les présenter de manière appropriée.
- Le concept de base des entrées et sorties est de gérer les interactions avec le programme durant son exécution. Elle permet de soumettre des informations au programme et de recevoir des données venant du programme.

2. Introduction aux Entrées/Sorties en Dart

1. **Introduction aux Entrées/Sorties en Dart**

   - Les entrées/sorties (E/S), souvent abrégées en "I/O" (pour Input/Output en anglais), sont une composante essentielle de la programmation.
   - Elles font référence à la manière dont un programme interagit avec le monde extérieur, que ce soit pour obtenir des données en entrée, afficher des résultats en sortie, ou communiquer avec d'autres périphériques ou systèmes.
   - Les E/S sont cruciales car elles permettent aux programmes de devenir interactifs et de résoudre des problèmes du monde réel.
   - Voici quelques-unes des raisons pour lesquelles les E/S sont importantes en programmation :

     1. **Interaction avec les utilisateurs :**

        - Les E/S permettent aux programmes de recevoir des informations directement des utilisateurs, que ce soit via le clavier, la souris, un écran tactile ou d'autres dispositifs d'entrée.
        - Cette interaction utilisateur est essentielle pour de nombreuses applications, notamment les logiciels graphiques, les applications de bureau, les jeux et les interfaces utilisateur web.

apcpedagogie.com

2. **Traitement de données externes :**
    - Les programmes doivent souvent lire des données à partir de sources externes, telles que des fichiers, des bases de données, des capteurs, des dispositifs de mesure, etc. Les E/S sont utilisées pour extraire ces données et les manipuler dans le programme.
3. **Affichage des résultats :**
    - Les programmes doivent également afficher les résultats de leurs calculs ou de leurs traitements. Cela peut se faire via l'affichage sur un écran, l'impression sur papier, l'envoi de données sur un réseau, ou d'autres moyens de communication.
4. **Communication entre processus :**
    - Les E/S sont utilisées pour la communication entre différents processus ou programmes. Par exemple, un programme peut communiquer avec une base de données, un service web ou d'autres systèmes externes pour obtenir des informations ou effectuer des opérations.
5. **Gestion des erreurs :**
    - Les E/S jouent un rôle clé dans la gestion des erreurs et des exceptions. Les programmes doivent être capables de détecter et de gérer les problèmes lorsqu'ils se produisent, que ce soit une erreur de lecture de fichier, une perte de connexion réseau, ou d'autres situations imprévues.
6. **Intégration matérielle :**
    - Les E/S sont essentielles pour interagir avec le matériel informatique, tels que les périphériques d'entrée/sortie, les disques durs, les imprimantes, les écrans, les caméras, etc.
    - Les pilotes et les interfaces de programmation sont utilisés pour gérer ces interactions matérielles.

2. **Aperçu des bibliothèques dart:io et http pour les opérations d'E/S.**

apcpedagogie.com

- Dart est un langage de programmation moderne développé par Google, principalement utilisé pour le développement d'applications mobiles, de serveurs et de web.
- Les bibliothèques dart:io et dart:io/http sont des parties intégrantes du SDK Dart, utilisées pour gérer les opérations d'entrées/sorties et les communications HTTP, respectivement.
- Voici un aperçu de ces deux bibliothèques :

1. **dart:io :**

   - Cette bibliothèque fournit des classes et des fonctions pour gérer les opérations d'entrées/sorties de bas niveau, telles que la lecture/écriture de fichiers, la création de sockets réseau, la gestion des processus, etc.
   - Vous pouvez l'utiliser pour créer des applications en ligne de commande, des serveurs, des scripts, etc.
   - Exemple de code pour la lecture d'un fichier :

```
import 'dart:io';

void main() {
  final File file = File('mon_fichier.txt');
  final contents = file.readAsStringSync();
  print(contents);
}
```

2. **dart:io/http :**

   - Cette bibliothèque est une extension de dart:io et est spécialement conçue pour effectuer des requêtes HTTP et gérer les réponses dans le contexte des applications serveur Dart.
   - Elle propose des classes telles que HttpClient et HttpServer pour créer des serveurs HTTP et effectuer des requêtes HTTP.
   - Exemple de code pour créer un serveur HTTP simple :

```
import 'dart:io';

void main() async {
  final server = await HttpServer.bind('localhost', 8080);
```

apcpedagogie.com

```
  print('Serveur écoutant sur ${server.address}:${server.port}');

  await for (HttpRequest request in server) {
    request.response
      ..write('Hello, World!')
      ..close();
  }
}
```

- En utilisant ces bibliothèques, vous pouvez créer des applications Dart robustes, notamment des serveurs web, des microservices, des clients HTTP, etc.
- Notez que Dart a également une bibliothèque standard de haut niveau appelée `dart:convert` qui vous permet de travailler avec des données JSON et d'autres formats courants, ce qui est souvent utile en conjonction avec `dart:io/http`.

3. Lecture et Écriture dans la Console

1. **Lecture de données depuis la console avec `stdin.readLineSync()`.**

   - En Dart, la méthode `stdin.readLineSync()` permet de lire des données depuis la console de manière synchrone (bloquante). Voici comment l'utiliser pour lire une ligne de texte depuis la console :

     1. **Exemple**

```
import 'dart:io';

void main() {
  print("Veuillez entrer une ligne de texte :");

  // Utilisez stdin.readLineSync() pour lire une ligne de texte
depuis la console.
  String? ligne = stdin.readLineSync();

  // Affichez la ligne lue depuis la console.
  print("Vous avez entré : $ligne");
}
```

     2. **Explications :**

- Nous importons la bibliothèque `dart:io` pour avoir accès à stdin et d'autres fonctionnalités d'E/S.
- Nous utilisons `print()` pour afficher un message à l'utilisateur, l'invitant à entrer une ligne de texte.
- Ensuite, nous utilisons `stdin.readLineSync()` pour lire une ligne de texte depuis la console. Cette fonction bloquera l'exécution du programme jusqu'à ce que l'utilisateur entre une ligne de texte et appuie sur la touche "Entrée".
- Nous stockons la ligne lue dans la variable ligne.
- Enfin, nous utilisons `print()` pour afficher la ligne lue depuis la console.
- Lorsque vous exécutez ce programme, il attendra que vous entriez une ligne de texte, puis affichera ce que vous avez entré.
- La méthode `readLineSync()` est utile pour créer des programmes interactifs en ligne de commande qui nécessitent des entrées de l'utilisateur. Elle renverra la ligne de texte sous forme de chaîne (String).
- Assurez-vous d'importer la bibliothèque `dart:io` et de l'exécuter en dehors de l'environnement web, car cette méthode n'est généralement pas utilisée pour les applications web, où les interactions avec l'utilisateur se font via des formulaires web.

- **La différence entre `String? ligne = stdin.readLineSync();` et `String ligne = stdin.readLineSync();`**
  - En Dart, la différence entre String? ligne = stdin.readLineSync(); et String ligne = stdin.readLineSync(); réside dans la gestion de la nullabilité des types, qui a été introduite avec la version Dart 2.12.
  - `String? ligne = stdin.readLineSync(); :`
    - Le type String? signifie que la variable ligne peut contenir une valeur de type String, mais elle peut également contenir null.
    - Cela signifie que si stdin.readLineSync() renvoie une valeur null, votre programme ne générera pas d'erreur et ligne contiendra simplement null.
  - `String ligne = stdin.readLineSync();:`
    - Dans ce cas, vous déclarez ligne comme une variable de type String, ce qui signifie qu'elle ne peut pas contenir null.

apcpedagogie.com

- Si stdin.readLineSync() renvoie null, vous obtiendrez une erreur (probablement une exception de type NullThrownError ou similaire) au moment de l'assignation à ligne.
- L'utilisation de String? est utile lorsque vous vous attendez à ce qu'une valeur puisse être nulle (par exemple, si la méthode stdin.readLineSync() peut retourner null dans certaines conditions) et que vous souhaitez gérer cette possibilité sans provoquer d'exceptions en cas de null.

- Pour résumer, String? permet de déclarer une variable qui peut contenir soit une chaîne de caractères valide, soit null, tandis que String déclare une variable qui ne peut contenir que des chaînes de caractères valides.

2. **Affichage de données dans la console avec `stdout.write()` et `print()`.**

- En Dart, vous pouvez afficher des données dans la console en utilisant les fonctions `stdout.write()` et `print()`. Voici comment les utiliser :
  - `stdout.write()`: Cette fonction permet d'afficher du texte dans la console sans saut de ligne à la fin.

```
import 'dart:io';

void main() {
  stdout.write("Ceci est affiché sans saut de ligne. ");
  stdout.write("Ceci est sur la même ligne. ");
  stdout.write("Et ceci aussi.");
}
```

  - L'exemple ci-dessus affiche le texte sans ajouter de sauts de ligne, de sorte que tout le texte s'affiche sur la même ligne.
  - `print()` : La fonction print() permet d'afficher du texte dans la console avec un saut de ligne à la fin.

```
import 'dart:io';

void main() {
  print("Ceci est affiché avec un saut de ligne.");
  print("Chaque appel à 'print' commence sur une nouvelle ligne.");
}
```

- L'exemple ci-dessus affiche le texte en ajoutant automatiquement un saut de ligne à la fin, ce qui signifie que chaque nouvel appel à print commence sur une nouvelle ligne.
- Il est important de noter que vous pouvez également utiliser des caractères de contrôle tels que "\n" pour ajouter des sauts de ligne avec `stdout.write()` si nécessaire, par exemple :

```
import 'dart:io';

void main() {
  stdout.write("Ceci est affiché sans saut de ligne.\n");
  stdout.write("Ceci est sur une nouvelle ligne.\n");
}
```

  - Cela ajoutera manuellement un saut de ligne en utilisant le caractère "`\n`". Vous pouvez choisir entre `stdout.write()` et `print()`+ en fonction de vos besoins spécifiques en matière de mise en forme de la sortie dans la console.

3. **Exercices pratiques pour saisir et afficher des données utilisateur.**

    1. **Exercice 1**

        - **Enoncé**
            - Demandez à l'utilisateur de saisir son nom.
            - Affichez "Bonjour, [nom de l'utilisateur] !" dans la console.
        - **Solution**

            Solution
            - Pour saisir un nom depuis l'utilisateur et l'afficher dans la console en Dart, vous pouvez utiliser la fonction stdin.readLineSync() pour obtenir l'entrée de l'utilisateur, puis utilisez `print()` pour afficher le message. Voici comment faire :

```
import 'dart:io';

void main() {
  print("Veuillez entrer votre nom :");

  // Utilisez stdin.readLineSync() pour lire la saisie de
```

```
l'utilisateur.
  String? nomUtilisateur = stdin.readLineSync();

  // Affichez un message de salutation en utilisant le nom de
l'utilisateur.
  print("Bonjour, $nomUtilisateur !");
}
```

- Nous importons la bibliothèque `dart:io` pour avoir accès à `stdin`.
- Nous utilisons `print()` pour afficher un message demandant à l'utilisateur de saisir son nom.
- La fonction `stdin.readLineSync()` est utilisée pour lire la saisie de l'utilisateur et la stocker dans la variable nomUtilisateur.
- Enfin, nous utilisons `print()` pour afficher le message de salutation en utilisant le nom de l'utilisateur saisi.
- Lorsque vous exécutez ce programme, il demandera à l'utilisateur de saisir son nom, puis affichera "Bonjour, [nom de l'utilisateur] !" en remplaçant [nom de l'utilisateur] par le nom saisi.

2. **Exercice 2**

   - **Enoncé**
     - Demandez à l'utilisateur de saisir trois notes (nombres à virgule flottante).
     - Calculez la moyenne de ces trois notes.
     - Affichez la moyenne dans la console.
   - **Solution**

     Solution
     - Pour créer un calculateur de moyenne en Dart, suivez ces étapes :

```
import 'dart:io';

void main() {
  // Demandez à l'utilisateur de saisir trois notes.
  double note1, note2, note3;

  print("Veuillez entrer la première note :");
  note1 = double.parse(stdin.readLineSync()!);

  print("Veuillez entrer la deuxième note :");
```

apcpedagogie.com

```
  note2 = double.parse(stdin.readLineSync()!);

  print("Veuillez entrer la troisième note :");
  note3 = double.parse(stdin.readLineSync()!);

  // Calculez la moyenne des trois notes.
  double moyenne = (note1 + note2 + note3) / 3;

  // Affichez la moyenne dans la console.
  print("La moyenne des trois notes est : $moyenne");
}
```

- **Explication**
  - Nous importons la bibliothèque dart:io pour avoir accès à stdin.
  - Nous demandons à l'utilisateur de saisir trois notes en utilisant `print()` et `stdin.readLineSync()`. Pour chaque note, nous utilisons `double.parse()` pour convertir la saisie de l'utilisateur en un nombre à virgule flottante (double).
  - Ensuite, nous calculons la moyenne en ajoutant les trois notes et en divisant le total par 3.
  - Enfin, nous utilisons `print()` pour afficher la moyenne dans la console.

3. **Exercice 3**

- **Enoncé**
  - Demandez à l'utilisateur de saisir une température en degrés Celsius.
  - Convertissez cette température en degrés Fahrenheit en utilisant la formule : Fahrenheit = (Celsius * 9/5) + 32.
  - Affichez la température en degrés Fahrenheit.
- **Solution**

  Solution
  - Pour créer un programme de conversion de température de degrés Celsius en degrés Fahrenheit en Dart, vous pouvez suivre ces étapes :

```
import 'dart:io';

void main() {
  // Demandez à l'utilisateur de saisir une température en degrés
Celsius.
```

apcpedagogie.com

```
  print("Veuillez entrer la température en degrés Celsius :");
  double celsius = double.parse(stdin.readLineSync());

  // Convertissez la température en degrés Fahrenheit en utilisant
la formule.
  double fahrenheit = (celsius * 9 / 5) + 32;

  // Affichez la température en degrés Fahrenheit.
  print("La température en degrés Fahrenheit est : $fahrenheit
°F");
}
```

- **Explication**
  - Nous importons la bibliothèque dart:io pour avoir accès à stdin.
  - Nous demandons à l'utilisateur de saisir une température en degrés Celsius en utilisant `print()` et stdin.readLineSync(). Nous utilisons double.parse() pour convertir la saisie de l'utilisateur en un nombre à virgule flottante (double).
  - Ensuite, nous convertissons la température en degrés Fahrenheit en utilisant la formule : Fahrenheit = (Celsius * 9/5) + 32.
  - Enfin, nous utilisons print() pour afficher la température en degrés Fahrenheit dans la console, en la concaténant avec le symbole "°F".

4. **Exercice 4**

- **Enoncé**
  - Demandez à l'utilisateur de saisir deux nombres et un opérateur (+, -, *, /).
  - Effectuez le calcul correspondant aux nombres et à l'opérateur saisis.
  - Affichez le résultat.
- **Solution**

  Solution
  - Pour créer une mini calculatrice en Dart qui demande à l'utilisateur de saisir deux nombres et un opérateur (+, -, *, /) et affiche le résultat, vous pouvez suivre ces étapes :

```
import 'dart:io';

void main() {
  // Demandez à l'utilisateur de saisir le premier nombre.
```

```
  print("Veuillez entrer le premier nombre :");
  double nombre1 = double.parse(stdin.readLineSync());

  // Demandez à l'utilisateur de saisir l'opérateur (+, -, *, /).
  print("Veuillez entrer l'opérateur (+, -, *, /) :");
  String operateur = stdin.readLineSync();

  // Demandez à l'utilisateur de saisir le deuxième nombre.
  print("Veuillez entrer le deuxième nombre :");
  double nombre2 = double.parse(stdin.readLineSync());

  double resultat;

  // Effectuez le calcul correspondant aux nombres et à
l'opérateur saisis.
  switch (operateur) {
    case "+":
      resultat = nombre1 + nombre2;
      break;
    case "-":
      resultat = nombre1 - nombre2;
      break;
    case "*":
      resultat = nombre1 * nombre2;
      break;
    case "/":
      if (nombre2 != 0) {
        resultat = nombre1 / nombre2;
      } else {
        print("Division par zéro impossible.");
        return;
      }
      break;
    default:
      print("Opérateur invalide.");
      return;
  }

  // Affichez le résultat.
  print("Résultat : $resultat");
}
```

- **Explication**
  - Nous importons la bibliothèque dart:io pour avoir accès à stdin.

apcpedagogie.com

- Nous demandons à l'utilisateur de saisir le premier nombre en utilisant print() et `stdin.readLineSync()`. Nous utilisons `double.parse()` pour convertir la saisie de l'utilisateur en un nombre à virgule flottante (double).
- Nous demandons à l'utilisateur de saisir l'opérateur (+, -, *, /).
- Nous demandons à l'utilisateur de saisir le deuxième nombre de la même manière que le premier.
- Nous utilisons une déclaration switch pour effectuer le calcul correspondant aux nombres et à l'opérateur saisis. Si l'utilisateur entre un opérateur invalide ou tente de diviser par zéro, nous affichons un message d'erreur.
- Enfin, nous utilisons print() pour afficher le résultat du calcul dans la console.

5. **Exercice 5**

   - **Enoncé**
     - Demandez à l'utilisateur de saisir une somme en euros.
     - Convertissez cette somme en dollars en utilisant un taux de change fixe.
     - Affichez la somme en dollars.
   - **Solution**

     Solution
     - Pour créer un programme en Dart qui permet à l'utilisateur de convertir une somme en euros en dollars en utilisant un taux de change fixe, suivez ces étapes :

```
import 'dart:io';

void main() {
  // Demandez à l'utilisateur de saisir une somme en euros.
  print("Veuillez entrer la somme en euros :");
  double euros = double.parse(stdin.readLineSync());

  // Définissez un taux de change fixe (par exemple, 1 euro = 1.18
dollars).
  double tauxDeChange = 1.18;

  // Convertissez la somme en dollars en utilisant le taux de
change.
```

apcpedagogie.com

```
  double dollars = euros * tauxDeChange;

  // Affichez la somme en dollars.
  print("La somme en dollars est : $dollars USD");
}
```

- **Explication**
  - Nous importons la bibliothèque dart:io pour avoir accès à stdin.
  - Nous demandons à l'utilisateur de saisir une somme en euros en utilisant print() et stdin.readLineSync(). Nous utilisons double.parse() pour convertir la saisie de l'utilisateur en un nombre à virgule flottante (double).
  - Nous définissons un taux de change fixe (par exemple, 1 euro = 1.18 dollars). Vous pouvez modifier cette valeur en fonction du taux de change actuel.
  - Nous utilisons le taux de change pour convertir la somme en euros en dollars en effectuant une simple opération de multiplication.
  - Enfin, nous utilisons `print()` pour afficher la somme en dollars dans la console, en la concaténant avec le symbole "USD".

6. **Exercice 6**

   - **Enoncé**
     - Générez un nombre aléatoire entre 1 et 100.
     - Demandez à l'utilisateur de deviner le nombre.
     - Donnez des indices (plus grand, plus petit, égal) jusqu'à ce que l'utilisateur trouve le nombre.
     - Affichez le nombre de tentatives nécessaires pour trouver la réponse.
   - **Solution**

     Solution
     - Pour créer un jeu de devinette en Dart, suivez ces étapes :

```
import 'dart:io';
import 'dart:math';

void main() {
  // Générez un nombre aléatoire entre 1 et 100.
  final int nombreSecret = Random().nextInt(100) + 1;
```

apcpedagogie.com

```
  int nombreEssais = 0;
  int devinette;

  print("Bienvenue dans le jeu de devinette !");
  print("Je pense à un nombre entre 1 et 100. Essayez de le
deviner.");

  while (true) {
    print("Veuillez entrer votre devinette :");
    devinette = int.parse(stdin.readLineSync());
    nombreEssais++;

    if (devinette == nombreSecret) {
      print("Félicitations ! Vous avez trouvé le nombre secret
($nombreSecret) en $nombreEssais essais.");
      break;
    } else if (devinette < nombreSecret) {
      print("Le nombre secret est plus grand.");
    } else {
      print("Le nombre secret est plus petit.");
    }
  }
}
```

- **Explication**
  - Nous importons les bibliothèques dart:io et dart:math.
  - Nous générons un nombre aléatoire entre 1 et 100 en utilisant Random().nextInt(100) + 1. Le nombre secret est stocké dans la variable nombreSecret.
  - Nous utilisons une boucle while (true) pour permettre à l'utilisateur de faire des devinettes jusqu'à ce qu'il trouve le nombre secret.
  - À chaque itération de la boucle, nous demandons à l'utilisateur de faire une devinette en utilisant stdin.readLineSync().
  - Nous comparons la devinette de l'utilisateur avec le nombre secret. Si la devinette est correcte, nous affichons un message de félicitations et le nombre d'essais. Sinon, nous donnons un indice en affichant "Le nombre secret est plus grand" ou "Le nombre secret est plus petit" en fonction de la devinette de l'utilisateur.

apcpedagogie.com

- Nous incrémentons le compteur nombreEssais à chaque devinette pour suivre le nombre total d'essais.

4. Requêtes HTTP en Dart

1. **Introduction aux requêtes HTTP avec la bibliothèque http.**

- En Dart, vous pouvez effectuer des requêtes HTTP pour interagir avec des services web et récupérer des données en utilisant la bibliothèque http. Cette bibliothèque vous permet d'envoyer des requêtes HTTP telles que des requêtes GET, POST, PUT, DELETE, etc. et de gérer les réponses HTTP. Voici une introduction aux requêtes HTTP en Dart avec la bibliothèque http :
- Importer la bibliothèque HTTP : Pour commencer, vous devez ajouter la dépendance http à votre fichier pubspec.yaml (le fichier de configuration de votre projet Dart). Voici comment vous pouvez le faire :

```
dependencies:
  http: ^0.13.3  # Utilisez la dernière version de la bibliothèque HTTP
```

- Une fois la dépendance ajoutée, exécutez pub get ou flutter pub get pour installer la bibliothèque.
- Importez la bibliothèque HTTP dans votre fichier Dart :

```
import 'package:http/http.dart' as http;
```

- Effectuer une requête HTTP GET : Voici comment effectuer une requête HTTP GET pour récupérer des données à partir d'une URL :

```
Future fetchData() async {
  final response = await http.get(Uri.parse('https://exemple.com/api/data'));

  if (response.statusCode == 200) {
    print('Réponse : ${response.body}');
  } else {
    print('Échec de la requête. Statut : ${response.statusCode}');
  }
}
```

- Effectuer d'autres types de requêtes : Vous pouvez également effectuer des requêtes POST, PUT, DELETE, etc., en utilisant les méthodes correspondantes

apcpedagogie.com

(post, put, delete, etc.) de l'objet http. Par exemple, pour effectuer une requête POST avec des données JSON :

```
import 'dart:convert';

Future postData() async {
  final Map data = {'key': 'value'};
  final String jsonData = json.encode(data);

  final response = await http.post(
    Uri.parse('https://exemple.com/api/post'),
    headers: {'Content-Type': 'application/json'},
    body: jsonData,
  );

  if (response.statusCode == 201) {
    print('Succès de la requête POST');
  } else {
    print('Échec de la requête POST. Statut : ${response.statusCode}');
  }
}
```

- Gérer les réponses HTTP : Vous pouvez extraire et analyser les données de la réponse HTTP en fonction du format de réponse (par exemple, JSON). Utilisez json.decode(response.body) pour analyser une réponse JSON.
- Gérer les erreurs : N'oubliez pas de gérer les erreurs qui peuvent survenir lors des requêtes HTTP, telles que des erreurs de réseau ou des erreurs de serveur.

2. **Utilisation de la bibliothèque http pour effectuer des requêtes GET et POST.**

3. **Traitement des réponses HTTP et affichage des données.**

4. **Exercices pratiques avec des API publiques.**

Cours Flutter

Source : https://apcpedagogie.com/les-entrees-et-sorties-en-dart · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com