Kiểu dữ liệu dart

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểu dữ liệu** | | **Nội dung** | |
| **Num** | | **bao gồm int và double** | |
| **Int (số)** | | khai báo rỗng khác với java (int ? a). | |
| **Double( số thập phân)** | | khai báo rỗng khác với java (double ? a). | |
| **Bool** | | Boolean. | |
| **String** | | chuỗi | |
| **Var** | | khai báo biến động khi không có giá trị mặc định ban đầu có thể gán bằng bất kỳ kiểu dữ liệu nào. Ngược lại khi có giá trị mặc định ban đầu thì phải theo kiểu dữ liệu giá trị mặc định đó. | |
| **Dynamic** | | khác với var có thể gán giá trị bằng bất kỳ giá trị nào kể cả khi có giá trị mặc định ban đầu hay không. | |
| **Cấu trúc dữ liệu** | | | |
| **enum** | | Giống obj trong java.  Cách khai báo : enum person {a, b, c}  Tên biến không được trùng với tên phần tử | |
| **Iterable** | | Liệt kê các phần tử dạng số | |
| **List** | **Không chỉ định kiểu dữ liệu** | Cách khai báo: var list1 = [];  Có thể add giá trị nhiều kiểu dữ liệu vào trong mảng | **//insert thêm giá trị vào mảng tại vị trí**  // list1.insert(4,3);    **// // xóa dữ liệu**  // // list1.remove(7);  **// // xóa dữ liệu tại vị trí**  // // list1.removeAt(4);  **// // xóa dữ liệu tại vị trí cuối**  // // list1.removeLast();  **// // xóa dữ liệu trong khoảng**  list1.removeRange(0,3);  **// // clear mảng**  // // list1.clear();    **reversed đảo chiều list**  list1.reversed.forEach((x) {  // print(x);  // }); |
| **Chỉ định kiểu dữ liệu** | Cách khai báo: List<kiểu dữ liệu> numbers = [];  Chỉ có thể add giá trị của kiểu dữ liệu được chỉ định. **Ngoại trừ kiểu dữ liệu Dynamic** |
| **Maps** | | Cách khai báo: var map1 = {};  Loại bỏ cáo phần tử trùng nhau  biểu diễn dưới dạng key value  //them phan tu  map['xa'] = 'dang xa';  // add 1 map khac vao  map1.addAll(map);  map1.addAll(map2);    //xoa phan tu  map1.remove('huyen');  // kiem tra phan tu map  bool check = map1.containsKey('huyen'); | |
| **Sets** | | Cách khai báo  var map1 = <kiểu dữ liệu>{}  Set <kiểu dữ liệu> set1 = {}  Loại bỏ cáo phần tử trùng nhau giống với maps  Khác với map, set chỉ có value không có key  // add phần tử  set1.add (1);  // add 1 set khác  set1.addAll(set); (chỉ khi có cùng kiểu dữ liệu mới có thể dùng addAll, ngoại trừ dynamic)  //xóa giá trị  set1.remove(<giá trị cần xóa>)  set1.removeAll(<giá trị cần xóa>)  // lấy phần tử bất kỳ  set.elementAt(1)  //check phần tử  Bool check = set.contains(‘<phần tử cần check>’) | |
| **Queue( hàng đợi)** | | Cách khai báo :  var q = Queue();  Queue<int> check = Queue();  var test = ['x', 'y', 'z'];  //thêm phần tử  q.add('a');  q.add('b');  q.add('c');    //thêm vào đầu hàng đợi  q.addFirst('e');  // thêm vào cuối hàng đợi  q.addLast('d');  // thêm 1 mảng vào hàng đợi  q.addAll(test);  // xóa giá trị  q.remove('e');  // xóa giá trị đầu  q.removeFirst();  // xóa giá trị cuối  q.removeLast(); | |
| **Chuyển đổi cấu trúc dữ liệu** | | Set<String> set1 = {};    //chuyển đổi từ kiểu này sang kiểu khác    //cách 1 : duyệt từng phần tử  print('cách 1 : duyệt từng phần tử');  lstNumber.forEach((x){  set1.add('$x');  });    set1.forEach((o){  print(o);  });      // cách 2 addAll  print('cách 2 addAll');  Set<String> set2 = {};  set2.addAll(lstString);    set2.forEach((o){  print(o);  });    // cách 3 .from  print('cách 3 .from');  Set<String> set3 = Set.from(lstString);    set3.forEach((o){  print(o);  });    // cách 4 .map()  print('cách 4 .map()');  List<String> strNumbers = lstNumber.map((x){  return '$x';  }).toList();    strNumbers.forEach((i){  print(i);  });    Set<String> setNumbers = lstNumber.map((x){  return '$x';  }).toSet();    setNumbers.forEach((i){  print(i);  });    } | |

|  |  |
| --- | --- |
| Toán tử dart | |
| **Kiểu** | **Nội dung** |
| //toán tử số học | print(a + b);  print(a - b);  print(a \* b);  print(a / b);  //lấy phần nguyên  print(a ~/ b);  //lấy phần dư  print(a % b);    //lấy trị tuyệt đối  print(c.abs()); |
| // toán tử kiểm tra kiểu as, is, is! | print(a is int); |
| //toán tử gán +=, -=, \*=, /= | // a += b <=> a = a + b;  // a \*= b <=> a = a \* b;  // a /= b <=> a = a / b;  // a -= b <=> a = a - b; |
| //toán tử logic | // giống java |
| //toán tử Bitwise và shift | google |
| //Biểu thức điều kiện | name = (check == null) ? 'a' : 'b';    name = check ?? 'c';  //cascades    List<int> numbers = [];    numbers..add(1)..add(3);  // =>  numbers.forEach((x) => print(x)); |

|  |  |
| --- | --- |
| Loop | |
| **Kiểu** | **Nội dung** |
| For | For(int i = 0 ; I < i.length; i++){} |
| For in | For(int number in numbers){} |
| forEach | numbers.forEach(x => print(x)) |
| while | While(1 < 2){  Print(‘1’)  } |
| Do while | Do{  Print(‘c’)  }while(1=1) |
| break | Dừng lặp |
| continue | Bỏ qua |

|  |  |
| --- | --- |
| **Function** | |
| **Kiểu** | **Nội dung** |
| Thông thường | Truyền param giống trong java |
| // optional tham số | void paramOptional(int a ,[String ? b, bool ? c]){  print(a);  print(b);  print(c);  }  Param trong ngoặc vuông khi gọi hàm có thể truyền hoặc không. Khi khai báo param cần khởi tạo chấp nhận kiểu dữ liệu null |
| // name parameter | void funcNameParameter({int? a, int? b, int? c}) {  print(a);  print(b);  print(c);  }  Khởi tạo param tương tự như **optional tham số** ta cần khởi tạo chấp nhận giá trị null, khác với **optional tham số**  thì ta để param trong **{}** , khi gọi hàm cần có tên param  cách gọi hàm và truyền param funcNameParameter(a : 1); |

|  |  |
| --- | --- |
| Lớp đối tượng và constructor | // khai báo thuộc tính của đối tượng int id; String name;  // can co ham khoi tao constructor khi khong gan tham so //constructor thuong // User(this.id, this.name); // name constructor // User.name(this.id, this.name); // name acument User({required this.id, this.name = 'dsada'});  @override String toString() {  // *TODO: implement toString* return '$id - $name'; }  Gọi class và truyền tham số  // constructor thuong User user = User(2, 'hahahahaha'); // name constructor User user = User.name(2, 'hahahahaha'); // name acument User user = User(id: 3); |
| Phạm vi truy suất và getter & setter | // khai báo thuộc tính của đối tượng private int \_id; String \_name;  //constructor  User(this.\_id, this.\_name);  String get name => \_name;  set name(String value) {  \_name = value; }  int get id => \_id;  set id(int value) {  \_id = value; } |
| Tính kế thừa | Constructor tính kế thừa  User(this.id, this.name) : super(24);  Lớp cha  class Person {  int age = 10;   void logAge(){  print('age');  }   Person(this.age); }  1 class con chỉ có thể kế thừa từ 1 class cha |
| Tính trừu tượng | Khi thế thừa 1 lớp trừu tượng cần override lại  hàm trừu tượng trong class đó  //lớp trừu tượng abstract class People {  //hàm trừu tượng  void func1(); } |
|  |  |
|  |  |