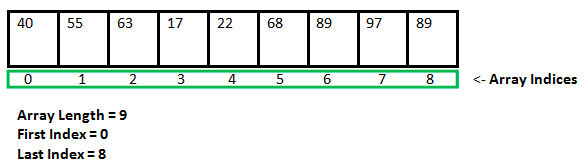
**[Bài đọc] Mảng một chiều**

*(Nguồn: https://www2.southeastern.edu/Academics/Faculty/kyang/2007/Fall/CMPS161/ClassNotes/CMPS161ClassNotesChap06.pdf)*

**Giới thiệu**

Mảng là một biến tham chiếu đến một loạt giá trị liên tiếp nhau. Các giá trị được lưu trữ trong mảng có cùng kiểu dữ liệu.



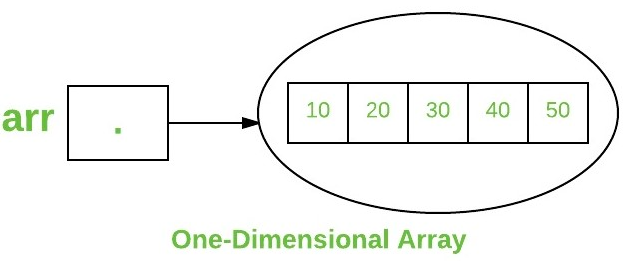
Thông thường bạn sẽ phải lưu trữ một số lượng lớn các giá trị trong quá trình thực hiện một chương trình. Giả sử bạn cần phải đọc 100 số, tính trung bình của chúng, và tìm ra có bao nhiêu con số trên mức trung bình. Chương trình của bạn lần đầu tiên đọc các con số và tính trung bình của chúng, sau đó so sánh mỗi con số với mức trung bình để xác định xem nó có ở trên mức trung bình hay không. Để hoàn thành nhiệm vụ này, tất cả các số phải được lưu trữ trong các biến. Bạn phải khai báo 100 biến và lặp lại nhiều lần viết mã gần như giống hệt nhau. Viết một chương trình theo cách này sẽ không thực tế. Vì vậy, làm thế nào để bạn giải quyết vấn đề này?

Một cách tiếp cận hiệu quả, có tổ chức là cần thiết. Java và hầu hết các ngôn ngữ cấp cao khác cung cấp một cấu trúc dữ liệu được gọi là mảng,  mà ở đó lưu trữ một tập các phần tử có cùng một kiểu dữ liệu. Trong ví dụ trên, bạn có thể lưu trữ tất cả 100 số vào một mảng và truy cập chúng thông qua một biến mảng.

Trong phần này giúp chúng ta tìm hiểu về mảng một chiều và các thao tác với mảng một chiều.

**Mảng một chiều**

Một khi mảng được tạo, kích thước của nó là cố định. Một biến tham chiếu mảng được sử dụng để truy cập các phần tử trong một mảng thông qua chỉ số mảng.



Một mảng được sử dụng để lưu trữ một tập dữ liệu, nhưng thường thì chúng ta thấy hữu ích hơn khi nghĩ về một mảng như một tập hợp các biến cùng loại. Thay vì khai báo từng biến riêng lẻ, chẳng hạn như number0, number1,. . . , và number99, bạn khai báo một biến mảng là numbers và sử dụng numbers[0], numbers[1], …. và numbers[99] để biểu diễn từng biến riêng lẻ.

**Khai báo mảng**

Để sử dụng một mảng trong một chương trình, bạn phải khai báo một biến để tham chiếu tới mảng và xác định kiểu dữ liệu của các phần tử trong mảng. Đây là cú pháp để khai báo một biến mảng:

elementType[] arrayRefVar;

**elementType**có thể là bất cứ kiểu dữ liệu nào (int, double, char, String, Object …) và tất cả các phần tử trong mảng phải có cùng một kiểu dữ liệu, **arrayRefVar** là tên biến mảng. Ví dụ, đoạn mã sau khai báo biến myList tham chiếu tới mảng các phần tử có kiểu dữ liệu số thực (double)

**double**[] myList;

***Chú ý:***

Bạn cũng có thể sử dụng cú pháp **elementType arrayRefVar[]**để khai báo một biến mảng. Cú pháp này xuất phát từ ngôn ngữ C / C ++ và đã được thông qua trong Java. Cú pháp **elementType[] arrayRefVar** vẫn được sử dụng nhiều hơn.

**Tạo mảng**

Không giống như các khai báo cho các biến kiểu dữ liệu nguyên thuỷ (primitive data type), khai báo biến mảng không cấp phát bất kỳ không gian nào trong bộ nhớ cho mảng đó. Nó tạo ra chỉ một vị trí lưu trữ để tham chiếu đến một mảng. Nếu một biến không chứa một tham chiếu đến một mảng, giá trị của biến là null. Bạn không thể gán các phần tử cho một mảng trừ khi nó đã được tạo ra. Sau khi một biến mảng được khai báo, bạn có thể tạo một mảng bằng cách sử dụng toán tử **new**và gán tham chiếu cho biến với cú pháp sau:

arrayRefVar = **new** elementType[arraySize]*;*

Câu lệnh này làm hai việc: (1) Tạo một mảng bằng cách sử dụng **new elementType[arraySize**; (2) Gán tham chiếu của mảng vừa được tạo ra cho biến arrayRefVar.

Khai báo biến mảng, tạo mảng, và gán tham chiếu của mảng vào biến có thể kết hợp trong một câu lệnh như sau:

elementType[] arrayRefVar = **new** elementType[arraySize];

Hoặc

elementType arrayRefVar[] = **new** elementType[arraySize];

Ví dụ:

**double**[] myList = **new double**[**10**];

Câu lệnh này khai báo một biến mảng myList, tạo ra một mảng gồm 10 phần tử kiểu số thực (double) và gán tham chiếu cho myList. Để gán giá trị cho các phần tử trong mảng, sử dụng cú pháp:

arrayRefVar[index] = value;

Ví dụ, đoạn mã sau dùng để khởi tạo các giá trị cho các phần tử trong mảng.

myList[**0**] = **5.6***;*

myList[**1**] = **4.5***;*

myList[**2**] = **3.3***;*

myList[**3**] = **13.2***;*

myList[**4**] = **4.0***;*

myList[**5**] = **34.33***;*

myList[**6**] = **34.0***;*

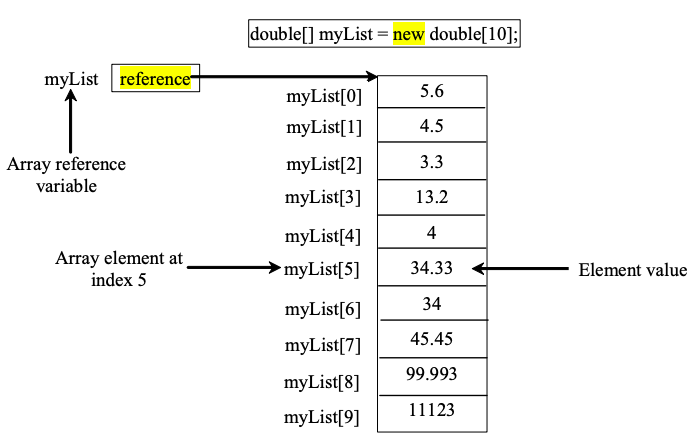
myList[**7**] = **45.45***;*

myList[**8**] = **99.993***;*

myList[**9**] = **11123***;*

Mảng này được minh hoạ trong hình 1.

**double**[] myList = **new double**[10];



Hình 1: Mảng myList gồm 10 phần tử có giá trị kiểu số thực (double) và số nguyên (int) và được đánh chỉ số từ 0 đến 0.

***Chú ý:***

Một biến mảng lưu trữ một mảng thực sự chứa một tham chiếu đến mảng đó. Nói đúng ra, một biến mảng và mảng là khác nhau, nhưng hầu hết sự phân biệt này có thể bị bỏ qua. Vì vậy, có thể nói chính xác rằng, myList là một mảng, myList là một biến có chứa một tham chiếu đến một mảng gồm mười phần tử kiểu số thực (double).

**Kích thước của mảng và giá trị mặc định**

Khi không gian cho một mảng được cấp phát, kích thước mảng phải được cung cấp để xác định số lượng các thành phần có thể được lưu giữ trong đó. Không thể thay đổi kích thước của mảng sau khi mảng được tạo. Kích thước có thể thu được bằng cách sử dụng **arrayRefVar.length**. Ví dụ: myList.length là 10.

Khi một mảng được tạo, các phần tử của nó được gán giá trị mặc định là 0 cho các kiểu dữ liệu số nguyên thủy, \ u0000 cho kiểu char, và false đối với các kiểu boolean.

**Truy xuất các phần tử trong mảng**

Các phần tử mảng được truy cập thông qua chỉ số. Chỉ số mảng bắt đầu từ 0, điều này có nghĩa là khoảng truy cập các phần tử trong mảng từ 0 đến arrayRefVar.length-1. Ví dụ trong Hình 1, myList chứa 10 giá trị double và chỉ số được đánh từ 0 đến 9.

Mỗi phần tử trong mảng được truy xuất thông qua chỉ số mảng với cú pháp như sau:

arrayRefVar[index];

Ví dụ, myList [9] biểu diễn cho phần tử cuối cùng trong mảng myList.

***Chú ý:***

Một số ngôn ngữ lập trình sử dụng dấu ngoặc đơn () để tham chiếu đến một phần tử mảng như **myList(9)**, nhưng trong Java sử dụng dấu ngoặc vuông [], như **myList[9]**.

Biến chỉ số được sử dụng giống như một biến thông thường. Ví dụ, đoạn mã sau cộng giá trị của phần tử myList[0] và myList[1] và gán cho phần tử myList[2]

myList[**2**] = myList[**0**] + myList[**1**]*;*

Sử dụng vòng lặp để gán giá trị cho các phần tử trong mảng **0**cho **myList[0]**, **1**cho **myList[1]**, . . . , và **9**cho **myList[9]**:

**for** (**int** i = **0**; i < myList.length; i++) {

myList[i] = i;

}

**Khởi tạo mảng**

Các giá trị trong mảng có thể được khởi tạo tại thời điểm khai báo. Cú pháp như sau:

elementType[] arrayRefVar = {value0, value1, ..., valuek};

Ví dụ:

**double**[] myList = {**1.9**, **2.9**, **3.4**, **3.5**};

khai báo, tạo và khởi tạo mảng myList gồm 4 phần tử, lệnh trên tương đương với lệnh khai báo sau:

**double**[] myList = **new double**[**4**];

myList[**0**] = **1.9**;

myList[**1**] = **2.9**;

myList[**2**] = **3.4**;

myList[**3**] = **3.5**;

***Chú ý:***

Toán tử **new** không được sử dụng trong cú pháp khởi tạo mảng. Khi khởi tạo các phần tử trong mảng, bạn cần phải theo thứ tự *khai báo*, *tạo* và *khởi tạo* trên cùng một câu lệnh. Nếu bỏ qua phần nào thì sinh lỗi cú pháp. Sau đây là một ví dụ sai về khởi tạo mảng:

**double**[] myList;

myList = {**1.9**, **2.9**, **3.4**, **3.5**};