| **So sánh** | **Array** | **Linked list** |  |
| --- | --- | --- | --- |
| Cơ bản | Là một tập cố định các phần tử | Một tập hợp lớn các dữ liệu được sắp xếp theo thứ tự nào đó |  |
| Kích thước | Cố định, được định nghĩa lúc khai báo | Không cần định nghĩa, nó tự lớn lên và thu gọn lại lúc thực thi |  |
| Nơi lưu trữ | Địa điểm của các thành phần được lưu trữ trong suốt thời gian compile | Vị trí của các thành phần được chỉ định ở run time |  |
| Thứ tự sắp xếp | Được lưu trữ liên tiếp nhau | Được lưu trữ ngẫu nhiên |  |
| Cách truy cập | Trực tiếp hoặc ngẫu nhiên | Truy cập một cách tuần tự |  |
| Chèn và xoá phần tử | Tương đối chậm do cần phải shifting mảng | Dễ dàng, nhanh chóng và tiện lợi |  |
| Tìm kiếm | Tìm kiếm nhị phân và tìm kiếm tuyến tính | Tìm kiếm tuyến tính |  |
| Bộ nhớ | Cần ít | Cần nhiều hơn |  |
| Sử dụng bộ nhớ | Không hiệu quả | Hiệu quả |  |

**Bảng so sánh giữa Array và List**

### **Vì sao ArrayList được sử dụng thay thế cho mảng?**

Array(mảng) và List được sử dụng trong quá trình phát triển ứng dụng Java, cả hai đều được sử dụng để lưu trữ dữ liệu nhưng tại sao chúng ta nên sử dụng List để thay thế cho mảng thông thường?

Trước hết, chúng ta cầm xem lại một vài nhược điểm điển hình của mảng thông thường trong Java:

* Kích thước mảng cố định và không thể thay đổi sau khi mảng được tạo.
* Không thể thêm phần tử sau khi mảng đã đầy.
* Việc thêm và xóa phần tử giữa mảng gây nhiều khó khăn.

Sau đây là những lý do nên dùng List:

* Kích thước của List có thể thay đổi tăng hoặc giảm tùy vào số lượng phần tử trong nó.
* Thêm và xóa phần tử ở bất kì vị trí nào trong mảng một cách nhanh chóng, hiệu quả.
* Có nhiều phương thức hỗ trợ các thao tác như removeAll-xóa tất cả, searching-tìm kiếm, iterations-duyệt mảng, retainAll-sửa phần tử,…
* Một List có thể chứa nhiều loại đối tượng như Integer, String, Scanner, hay một đối tượng do người dùng tự khai báo.

**Mảng (Array) và Danh sách (List) là cấu trúc dữ liệu phổ biến trong lập trình và có nhiều ứng dụng thực tế trong việc xử lý dữ liệu. Dưới đây là một số ví dụ về cách chúng được sử dụng trong thực tế:**

\*\*Ứng dụng của Mảng (Array):\*\*

1. \*\*Quản lý dữ liệu trong danh sách:\*\* Mảng thường được sử dụng để lưu trữ danh sách các phần tử cùng kiểu dữ liệu. Ví dụ, bạn có thể sử dụng mảng để lưu trữ danh sách sinh viên trong một lớp học.

2. \*\*Xử lý hình ảnh và âm thanh:\*\* Trong lập trình đồ họa và xử lý hình ảnh, mảng thường được sử dụng để biểu diễn các pixel hoặc mẫu âm thanh. Điều này cho phép bạn thực hiện các phép tính phức tạp trên hình ảnh hoặc âm thanh.

3. \*\*Sắp xếp và tìm kiếm:\*\* Mảng là một cấu trúc dữ liệu hiệu quả để thực hiện các thuật toán sắp xếp và tìm kiếm. Ví dụ, thuật toán QuickSort và Binary Search thường được thực hiện trên mảng.

4. \*\*Xây dựng ma trận và bảng:\*\* Trong đồ họa máy tính và tính toán khoa học, mảng được sử dụng để biểu diễn ma trận và bảng dữ liệu. Điều này hữu ích cho việc thực hiện các phép toán ma trận và phân tích dữ liệu.

\*\*Ứng dụng của Danh sách (List):\*\*

1. \*\*Quản lý danh bạ điện thoại:\*\* Danh sách có thể được sử dụng để lưu trữ danh bạ điện thoại, với mỗi liên hệ là một phần tử trong danh sách.

2. \*\*Lịch trình và quản lý công việc:\*\* Danh sách có thể được sử dụng để biểu diễn lịch trình hàng ngày hoặc danh sách công việc cần hoàn thành.

3. \*\*Lưu trữ lịch sử hoạt động:\*\* Danh sách có thể được sử dụng để lưu trữ lịch sử hoạt động của người dùng trên một ứng dụng hoặc trang web, như các bài đăng trên mạng xã hội.

4. \*\*Thực hiện hàng đợi (Queue) và ngăn xếp (Stack):\*\* Danh sách có thể được sử dụng để triển khai hàng đợi (Queue) hoặc ngăn xếp (Stack) trong các thuật toán và ứng dụng như quản lý tiến trình, duyệt các nút trong cây, và nhiều tình huống khác.

5. \*\*Lưu trữ dữ liệu động:\*\* Danh sách có kích thước động, có nghĩa là bạn có thể thêm hoặc loại bỏ phần tử dễ dàng, điều này hữu ích khi bạn cần lưu trữ dữ liệu mà số lượng phần tử có thể thay đổi theo thời gian.

**Tóm lại, mảng và danh sách đều là cấu trúc dữ liệu quan trọng và có nhiều ứng dụng trong lập trình và xử lý dữ liệu. Sự lựa chọn giữa chúng thường phụ thuộc vào yêu cầu cụ thể của dự án và tính hiệu suất mong muốn.**