# Memory

# 1. Thế nào là cấp phát tĩnh và cấp phát động

Cấp phát tĩnh là quá trình cấp phát bộ nhớ cho các biến hoặc cấu trúc dữ liệu tại thời điểm biên dịch (compile-time). Điều này có nghĩa là kích thước của biến hoặc đối tượng được xác định trước và không thể thay đổi trong suốt vòng đời của chương trình.

#### Đặc điểm của cấp phát tĩnh:

* Bộ nhớ được cấp phát tại thời điểm biên dịch, trước khi chương trình chạy.
* Kích thước cố định: Kích thước của các biến được quyết định tại thời điểm biên dịch và không thể thay đổi sau đó.
* Vùng nhớ sử dụng: Bộ nhớ được cấp phát tĩnh thường được lưu trữ trong các vùng nhớ như stack hoặc data segment (đối với các biến toàn cục).
* Không cần giải phóng: Vì bộ nhớ được giải phóng tự động khi chương trình kết thúc hoặc khi hàm hoàn tất.

Ưu điểm của cấp phát tĩnh:

* Đơn giản, nhanh chóng vì không cần thực hiện tại thời gian chạy (runtime).
* Không có khả năng gây ra lỗi rò rỉ bộ nhớ (memory leak).

Nhược điểm:

* Không linh hoạt vì kích thước bộ nhớ phải được xác định trước và không thể thay đổi trong quá trình thực thi chương trình.
* Sử dụng bộ nhớ không hiệu quả nếu không biết trước kích thước chính xác của dữ liệu.

Cấp phát động là quá trình cấp phát bộ nhớ tại thời gian chạy (runtime). Khi chương trình cần một vùng nhớ cho một biến hoặc đối tượng, nó sẽ yêu cầu hệ điều hành cấp phát một vùng nhớ thích hợp. Kích thước của bộ nhớ có thể được xác định linh hoạt dựa trên nhu cầu của chương trình trong quá trình chạy.

#### Đặc điểm của cấp phát động:

* Bộ nhớ được cấp phát tại thời gian chạy.
* Kích thước linh hoạt: Kích thước bộ nhớ có thể được thay đổi khi cần thiết trong thời gian chạy.
* Vùng nhớ sử dụng: Bộ nhớ được cấp phát động thường nằm trong heap.
* Yêu cầu giải phóng: Khi không còn sử dụng, bộ nhớ cấp phát động cần được giải phóng thủ công (như trong C/C++) hoặc được quản lý bởi trình thu gom rác (garbage collector) trong các ngôn ngữ như Java.

Ưu điểm của cấp phát động:

* Linh hoạt: Có thể cấp phát và giải phóng bộ nhớ tại thời gian chạy tùy theo nhu cầu.
* Hiệu quả hơn về mặt bộ nhớ: Giúp sử dụng bộ nhớ hiệu quả hơn khi kích thước của dữ liệu không xác định trước.

Nhược điểm:

* Chậm hơn: Quá trình cấp phát và giải phóng bộ nhớ tại runtime thường chậm hơn so với cấp phát tĩnh.
* Rủi ro rò rỉ bộ nhớ: Nếu bộ nhớ được cấp phát không được giải phóng đúng cách, có thể dẫn đến lỗi memory leak.

# 2. Phân biệt bộ nhớ heap và bộ nhớ stack ?

### **1. Bộ nhớ Stack**

Bộ nhớ **stack** là khu vực bộ nhớ được cấp phát tự động cho các biến cục bộ (local variables) và các lời gọi hàm trong một chương trình. Nó có cơ chế quản lý dữ liệu theo nguyên tắc **LIFO (Last In, First Out)**, tức là dữ liệu được đưa vào sau cùng sẽ được lấy ra trước tiên.

#### **Đặc điểm của bộ nhớ stack:**

* **Tự động cấp phát và giải phóng**: Khi một hàm được gọi, bộ nhớ cho các biến cục bộ của hàm đó sẽ được cấp phát trên stack. Khi hàm kết thúc, bộ nhớ này sẽ tự động được giải phóng.
* **Kích thước cố định**: Stack có kích thước cố định và giới hạn (phụ thuộc vào hệ thống hoặc trình biên dịch).
* **Tốc độ truy cập nhanh**: Do cách quản lý dữ liệu theo kiểu LIFO, truy cập vào bộ nhớ stack rất nhanh chóng.
* **Quản lý tự động**: Trình biên dịch tự động quản lý việc cấp phát và giải phóng bộ nhớ cho stack, không yêu cầu lập trình viên can thiệp.
* **Vùng lưu trữ cho các biến cục bộ**: Stack chủ yếu lưu trữ các biến cục bộ, các tham số của hàm và địa chỉ trả về của hàm.

**Ưu điểm của bộ nhớ stack:**

* Tốc độ truy cập nhanh do cách quản lý theo LIFO.
* Tự động quản lý bộ nhớ, không có nguy cơ rò rỉ bộ nhớ (memory leak).

**Nhược điểm:**

* Kích thước của stack bị giới hạn, dẫn đến lỗi **StackOverflowError** nếu quá nhiều bộ nhớ được cấp phát cho stack (thường xảy ra trong trường hợp đệ quy quá sâu).
* Không phù hợp để lưu trữ các đối tượng lớn hoặc các cấu trúc dữ liệu phức tạp.

### **2. Bộ nhớ Heap**

Bộ nhớ **heap** là khu vực bộ nhớ được sử dụng để cấp phát bộ nhớ **động (dynamic allocation)**, tức là khi chương trình chạy, bộ nhớ có thể được cấp phát và giải phóng tùy ý. Heap có cấu trúc không tổ chức rõ ràng như stack và phải được quản lý bởi lập trình viên hoặc cơ chế thu gom rác (garbage collector) trong các ngôn ngữ như Java.

#### **Đặc điểm của bộ nhớ heap:**

* **Cấp phát động**: Bộ nhớ heap được cấp phát khi chương trình yêu cầu (tại runtime) và phải được giải phóng thủ công (trong C/C++) hoặc tự động thông qua garbage collector (trong Java).
* **Kích thước linh hoạt**: Heap không có giới hạn chặt chẽ về kích thước như stack, có thể chứa nhiều đối tượng lớn.
* **Tốc độ truy cập chậm hơn**: Truy cập vào bộ nhớ heap chậm hơn so với stack do sự phức tạp trong việc quản lý và sắp xếp dữ liệu.
* **Lưu trữ đối tượng và biến toàn cục**: Bộ nhớ heap chủ yếu được sử dụng để lưu trữ các đối tượng được tạo ra trong quá trình chạy, đặc biệt khi kích thước của chúng không thể xác định trước.

**Ưu điểm của bộ nhớ heap:**

* Kích thước linh hoạt, có thể chứa các đối tượng lớn hoặc các cấu trúc dữ liệu phức tạp.
* Thích hợp để lưu trữ các đối tượng hoặc biến mà thời gian sống của chúng không thể xác định trước.

**Nhược điểm:**

* Truy cập vào heap chậm hơn so với stack.
* Nếu không quản lý tốt, có thể gây ra lỗi rò rỉ bộ nhớ (memory leak), đặc biệt trong các ngôn ngữ như C/C++.
* Cần cơ chế quản lý bộ nhớ, như **garbage collector** trong Java.

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Bộ nhớ Stack | Bộ nhớ Heap |
| Thời gian cấp phát | Tại thời điểm biên dịch (compile-time) | Tại thời gian chạy (runtime |
| Quản lý bộ nhớ | Tự động quản lý (trình biên dịch) | Lập trình viên hoặc garbage collector quản lý |
| Tốc độ | Nhanh hơn do cách quản lý LIFO | Chậm hơn do việc cấp phát và giải phóng không có cấu trúc cụ thể |
| Kích thước | Kích thước cố định, bị giới hạn | Kích thước linh hoạt, có thể mở rộng |
| Vị trí lưu trữ | Lưu các biến cục bộ, tham số hàm và địa chỉ trả về của hàm | Lưu trữ các đối tượng động, cấu trúc dữ liệu lớn, biến toàn cục |
| Quản lý vòng đời | Bộ nhớ được giải phóng tự động khi thoát khỏi phạm vi (scope) của hàm hoặc khối mã | Bộ nhớ cần được giải phóng thủ công hoặc tự động qua garbage collector |
| Lỗi tiềm ẩn | Có thể gây ra lỗi **StackOverflowError** khi cấp phát quá nhiều bộ nhớ | Có thể gây ra lỗi **Memory Leak** nếu không giải phóng bộ nhớ đúng cách |