**<a>Recursion – Phần 1</a>**

<p>Trong lập trình, recursion thường hay được bắt gặp trong hàm, nó không phải là một khái niệm xa lạ và mang tính ứng dụng rất cao, vậy recrusion là gì và khi nào thì nên sử dụng ?</p>

<p>Recursion dịch ra có nghĩa là lặp lại một quá trình hay một hoạt động, việc lặp lại này sẽ cho ra kết quả y hệt với quá trình hay hoạt động được lặp</p>

<p>Hãy cùng xem xét một ví dụ đơn giản về recrusion, viết một hàm tính giai thừa của 1 số bất kỳ</p>

<p>Theo cách làm thông thường, bài toán trên có thể giải quyết dễ dàng bằng vòng lặp i, vòng lặp này sẽ xuất phát từ số cần tính giai thừa sau đó giảm dần 1 đơn vị cho đến khi lớn hơn 1 thì dừng lại</p>

<p>function factorial(num) {</p>

<p> var fac = 1</p>

<p> **for (var i = num; i > 1; i--)** {</p>

<p> fac = fac \* i</p>

<p> }</p>

<p> return fac</p>

<p>}</p>

<p>Khi sử dụng phương thức recursion thì hàm tính giai thừa sẽ được lặp lại trong hàm tính giai thừa, việc lặp lại này có một đặc điểm là input của nó sẽ thay đổi bằng cách giảm một đợn vị sau mỗi lần được gọi lại, và khi input giảm đến bằng 1 thì nó sẽ trả kết quả bằng 1 và return toàn bộ kết quả</p>

<p>function **factorial(num)** {</p>

<p> if (num == 1) return 1; </p>

<p> return num \* **factorial(num-1)** </p>

<p>}</p>

<p>factorial(5) </p>

<p>Hai cách làm trên đều cho ra một kết quả tương tự, vậy đâu là điểm khác biệt giữa hai phương pháp trên? Để hiểu rõ về điều này trước tiên cần biết về khái niệm Call stack</p>

**<p><b>Call stack là gì ?</b></p>**

<p>Stack có nghĩa tập hợp nhiều thứ được xếp chồng lên nhau một cách ngăn nắp, tưởng tưởng bạn có một xấp tài liệu được xếp chồng lên nhau, những lá bài được xếp chồng lên nhau, hay những cuốn sách được đặt lên nhau thành chồng</p>

<p>Vậy call stack là gì? từ call có nghĩa là chỉ định, gọi hay triệu tập một điều gì đó. Tưởng tượng bạn có một chồng sách trên bàn và bạn muốn đọc sách, theo phản xạ tự nhiên bạn sẽ lấy ngay cuốn sách trên đầu chồng sách để xem trước hành động này có thể được xem là Call stack</p>

**<p><b>Vậy Call stack trong lập trình là gì ?</b></p>**

<p>Trong hầu hết ngôn ngữ lập trình, việc xây dựng cấu trúc dữ liệu là việc quản lý những gì xảy ra trong hàm khi chúng được gọi</p>

<p>Khi một hàm hoạt động, các sự kiện trong hàm sẽ bắt đầu quá trình thực thi, các sự kiện sẽ được đẩy lên phía trên bằng cách xếp chồng lên nhau được gọi là Call stack, và khi một sự kiện phía trên cùng của call stack hoàn thành, nó sẽ được xóa khỏi call stack, quá trình trên sẽ kết thúc khi không còn sự kiện nào trên Call stack. Để dễ tưởng tượng hãy xem những sự kiện trong hàm như các cuốn sách được xếp chồng lên nhau và bạn sẽ lấy cuốn sách trên cùng đọc trước, sau đó tiếp tục lấy các cuốn còn lại từ trên xuống để đọc, khi đọc xong cuốn sách cuối cùng trong chồng sách thì bạn đã thực hiện xong một công việc</p>

<p>Quay lại ví dụ ở đầu bài để xem Call stack hoạt động như thế nào</p>

<p>Viết một hàm tính giai thừa với đầu vào là 1 số ‘num’ bất kỳ, cụ thể mình sẽ cho num = 5 làm đầu vào</p>

<p>Để tính giai thừa của năm thì công việc sẽ gồm các bước sau: </p>

<p>B1: Đầu tiên gán giá trị của giai thừa là 1, factorial = 1, </p>

<p>B2: Tính giai thừa của 5, như vậy quá trình tính toán sẽ bắt đầu từ số 5 nhân cho giá trị giai thừa đã xác định trước đó là 1, khi đó factorial sẽ trở thành factorial = 1\*5 = 5, </p>

<p>B3: Tiếp tục thực hiện giống B2 bằng việc giảm số vừa được nhân đi 1 và nhân cho giá trị giai thừa đã xác định trước đó: factorial = 5\*(5-1), </p>

<p>B4: Lập lại quá trình giống B3 thu được factorial = 5\*4\*(4-1), </p>

<p>B5: Lập lại quá trình giống B3 thu được factorial = 5\*4\*3\*(3-1), </p>

<p>B6: trả về giá trị factorial = 5\*4\*3\*2 = 120, kết thúc quá trình tính toán</p>

<p>Toàn bộ các bước trên khi chuyển vào function factorial(5) được thực hiện như sau: </p>

<p>Đầu tiên Call stack sẽ thực hiện việc gọi hàm factorial(5) để thực hiện tính giai thừa cho 5, phần call stack bên phía phải màn hình hiển thị Call stack dòng 19</p>

**<div class="a">**

**<img src="pics/1.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Để thực hiện stack dòng 19 cần thực hiện stack dòng 13 trước tiên bằng việc gán var fac = 1. Lúc này stack dòng 13 sẽ chồng lên stack dòng 19 ở phần Call stack (hiển thị bên phải phía dưới hình) </p>

**<div class="a">**

**<img src="pics/2.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Khi stack dòng 13 hoàn thành, nó được gỡ bỏ và tiếp tục Call stack dòng 14, stack 14 sẽ thực hiện 2 bước là gán I = num và kiểm tra i > 1</p>

**<div class="a">**

**<img src="pics/3.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Khi stack dòng 14 hoàn thành, nó được gỡ bỏ và tiếp tục Call stack dòng 15, stack dòng 15 sẽ thực hiện nhiệm vụ tính tích của giá trị i và fac</p>

**<div class="a">**

**<img src="pics/4.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Stack dòng 15 được gỡ bỏ sau khi hoàn thành, vòng lặp sẽ quay lại stack dòng 14 và giảm i xuống 1 đơn vị và kiểm tra I > 1</p>

**<div class="a">**

**<img src="pics/5.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Gỡ bỏ stack dòng 14 sau khi hoàn thành, và tiếp tục thực hiện stack dòng 15</p>

**<div class="a">**

**<img src="pics/6.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Các bước trên sẽ được thực hiện lặp lại theo điều kiện vòng lặp và khi I giảm xuống còn 1 nó sẽ không thỏa mãn được stack dòng 14 (điều kiện I > 1), lúc này function chuyển sang stack dòng 17</p>

**<div class="a">**

**<img src="pics/7.png " alt="Trulli" width="700" height="500">**

**</div>**

<p>Sau khi stack dòng 17 thực hiện xong và trả về giá trị fac, function kết thúc đồng nghĩa stack dòng 19 cũng đã hoàn thành, lúc này stack dòng 17 & 19 sẽ được gỡ ra khỏi Call stack</p>

<p>Việc chạy function trên đã giúp bạn hiểu về Call stack. Nó cũng giống như bạn được giáo viên giao nhiệm vụ đọc sách, tưởng tượng việc tính giai thừa của số 5 cũng giống như việc đọc 5 cuốn sách trên kệ thư viện, bạn gọi (call) quyển sách số 5 trên kệ và nhân viên thư viện mang lại và đặt trên bàn cho bạn đọc, khi đọc xong quyển số 5 có nghĩa là bạn đã hoàn thành 1 stack, bạn bỏ cuốn sách ra khỏi bàn đọc và tiếp tục gọi nhân viên thư viện mang quyển sách số 4 đến cho bạn, bạn đọc hết quyển số 4 coi như hoàn thành tiếp được 1 stack và bạn lại bỏ cuốn sách số 4 ra khỏi bàn đọc. Quá trình trên cứ tiếp diễn như vậy cho đến khi bạn đọc xong quyển sách cuối cùng, lúc này bạn hoàn thành nhiệm vụ đọc sách</p>

<p>Khi đã hiểu Call stack, đây là cầu nối để hiểu về cách hoạt động của Recursion, phần 2 sẽ trình bày chi tiết và cách hoạt động của Recursion</p>