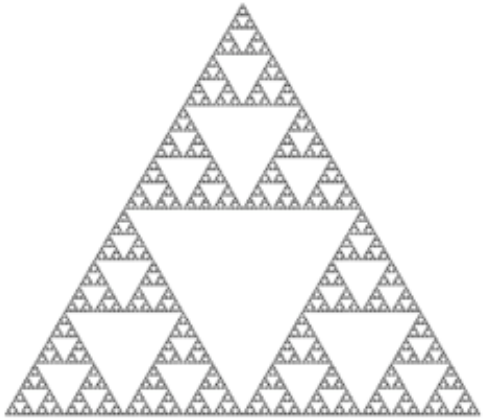
Đệ quy

Bởi:

Wiki Pedia

Tam giác Sierpinski



Đệ quy (tiếng Anh: recursion) là phương pháp dùng trong các chương trình máy tính trong đó có một hàm tự gọi chính nó.

Khái niệm hình thức về đệ quy

Trong toán học và khoa học máy tính, các tính chất (hoặc cấu trúc) được gọi là đệ quy nếu trong đó một lớp các đối tượng hoặc phương pháp được xác định bằng việc xác định một số rất ít các trường hợp hoặc phương pháp đơn giản (thông thường chỉ một) và sau đó xác định quy tắc đưa các trường hợp phức tạp về các trường hợp đơn giản.

Chẳng hạn, định nghĩa sau là định nghĩa đệ quy của tổ tiên:

- Bố mẹ của một người là tổ tiên của người ấy (trường hợp cơ bản);
- Bố mẹ của tổ tiên một người bất kỳ là tổ tiên của người ấy (bước đệ quy).

Các định nghĩa kiểu như vậy cũng thường thấy trong toán học (chính là quy nạp toán học)

Định nghĩa theo đệ quy

Một khái niệm X được định nghĩa theo đệ quy nếu trong định nghĩa X có sử dụng ngay chính khái niệm X.

Định nghĩa số tự nhiên

- 0 là một số tự nhiên.
- n là số tự nhiên nếu n - 1 là số tự nhiên.

Định nghĩa Hàm giai thừa n!

- 0! = 1
- Nếu n > 0 thì n! = n(n - 1)!

Đệ quy trong khoa học máy tính

Có một phương pháp chung để giải các bài toán là chia bài toán thành các bài toán con đơn giản hơn cùng loại. Phương pháp này được gọi là kỹ thuật lập trình chia để trị. Chính nó là chìa khóa để thiết kế nhiều giải thuật quan trọng, là cơ sở của quy hoạch động.

Một ví dụ cổ điển của đệ quy là hàm giai thừa cho bằng giả mã trong C hoặc C++ sau đây:

```
function factorial(n) { if (n <= 1) return 1; else return
n * factorial(n-1); }
```

Một ví dụ khác của giải thuật đệ quy là thủ tục duyệt (nghĩa là thực hiện một công việc náo đó với chúng) tất cả các nút của một cấu trúc dữ liệu cây:

```
procedure ProcessTree(node) { ProcessNode(node); // thực
hiện một thuật toán riêng với nút đầu for each child_node
of node do ProcessTree(child_node); }
```

Để duyệt một cây, gọi thủ tục này với nút gốc của cây như một tham biến khởi tạo. Tiếp theo, thủ tục gọi đệ quy đến chính nó cho tất cả các nút con của nút vừa gọi (nghĩa là các cây con của cây vừa gọi), cho đên khi gặp trường hợp cơ bản nghĩa là nút không có con (thường gọi là "lá").

Chính cấu trúc cây cũng được định nghĩa bằng đệ quy như sau:

```
struct node { child_nodes : list<node>; ... } struct tree
{ root : node; ... }
```

Cây được biểu diễn bằng một nút gốc và danh sách các nút con của nút ấy. Mỗi nút con lại có danh sách các nút con của nó (và như vậy, nó là gốc của một cây con). Lá với danh sách rỗng các nút con là trường hợp cơ sở của nút.

Chương trình con đệ qui

Trong lập trình, có khái niệm: một chương trình con (hàm, thủ tục) được gọi là đệ qui nếu trong quá trình thực hiện nó có phần phải gọi đến chính nó.

Cấu trúc chính

Một chương trình con đệ qui căn bản gồm hai phần.

- Phần cơ sở: chứa các tác động của hàm hoặc thủ tục với một số giá trị cụ thể ban đầu của tham số.
- Phần đệ qui: định nghĩa tác động cần được thực hiện cho giá trị hiện thời của các tham số bằng các tác động đã được định nghĩa trước đây với kích thước tham số nhỏ hơn.

Hàm tính giai thừa của một số tự nhiên n (tính n!) (Đoạn mã sau được viết bằng ngôn ngữ Pascal)

```
function gt(n: Word): Longint; begin if n = 1 then gt := 1
else gt := n * gt(n - 1); end;
```

Qui trình thực hiện

Trong ví dụ trên, qui trình thực hiện như sau:

- Khi có lệnh gọi hàm, chẳng hạn:

```
x := gt(3);
```

- thì máy sẽ ghi nhớ là:

```
gt(3) := 3 * gt(2);
```

và đi tính gt(2)

- kế tiếp máy lại ghi nhớ:

```
gt(2) := 2 * gt(1);
```

và đi tính gt(1)

- Theo định nghĩa của hàm thì:

```
gt(1) := 1;
```

- Máy sẽ quay ngược lại:

```
gt(2) := 2 * 1;
```

cho kết quả là 2

- Tiếp tục:

```
gt(3) := 3 * 2;
```

cho kết quả là 6

- Như vậy kết quả cuối cùng trả về là 6. Ta có: 3! = 6.

Đệ qui tương hỗ

Nếu có hai chương trình con B1 và B2 gọi lẫn nhau ta có đệ qui tương hỗ.