مبحث نهم

ساختمان داده ها

تابع بازگشتی

- تابع بازگشتی تابعی است که در بدنه اش دستوری دارد که خودش را فراخوانی می کند. توابع بازگشتی برای نگهداری حالت قبلی خود از پشته مکرر استفاده می کنند.

- مسئله ای که به صورت بازگشتی حل می شود باید بتواند به مسائل کوچک تر تقسیم بشود و حل مسائل کوچک به همان روش مسئله بزرگ قابل انجام باشد. مسئله کوچک تر به مسئله کوچک تری شکسته می شود تا سرانجام یه کوچک ترین اندازه مسئله برسد که base case نامیده می شود که می تواند بدون استفاده از بازگشتی حل شود.

تابع فاکتوریل

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times f(n-1) & \text{if } n > 0 \end{cases}$$

- مقدار تابع برای f(3)به صورت زیر محاسبه می شود:

f(3) = 3 * f(3 – 1)
= 3 * f(2)
= 3 * 2 * f(2 – 1)
= 3 * 2 * f(1)
= 3 * 2 * 1 * f(1 – 1)
= 3 * 2 * 1 * f(0)
= 3 * 2 * 1 * 1
= 6

تابع بازگشتی محاسبه فاکتوریل یک عدد صحیح

- ```
  int Factorial (int x)
  {
   if (x<= 1)
     return 1;
     return x * Factorial (x-1);
  }
  ```

- همین تابع به صورت غیر بازگشتی به صورت زیر نوشته می شود. توجه کنید که به د و متغیر موقت نیاز دارد.

- ```
  int Factorial (int x)
  {
   int i, temp;
   for ( i=1; i<=x; i++)
     temp*=i;
   return temp;
  }
  ```

بازگشتی در مقایسه با غیر بازگشتی

- برای اینکه تابع بازگشتی موفق باشد نیاز است مسئله یک زیرساختار بازگشتی داشته باشد. راه حل بعضی از مسائل به طور ذاتی بازگشتی است چون احتیاج به نگهداری حالت قبلی دارند.
- الگوریتم پیمایش درخت(tree traversal)، تابع اکرمن(Ackermann) و الگوریتم های تقسیم و غلبه مانند مرتب سازی سریع(Quicksort) همگی به صورت بازگشتی هستند. همه این الگوریتم ها می توانند به صورت غیربازگشتی با کمک پشته هم پیاده شوند اما نیاز به پشته مزیت راه حل غیر بازگشتی را از بین می برد.
- تابع غیر بازگشتی احتمالا در عمل کمی سریعتر از نسخه بازگشتی آن اجرا می شود چون تابع غیر بازگشتی سربار فراخوانی تابع (function-call) را به اندازه تابع بازگشتی ندارد و این سربار در بعضی زبان ها نسبتا بالا است.
- یک دلیل دیگر برای ترجیح غیربازگشتی به بازگشتی این است که فضای پشته قابل دسترس کمتر از فضای قابل دسترس در حافظه آزاد heap است. و الگوریتم های بازگشتی تمایل به فضای پشته بیشتری نسبت به غیر بازگشتی دارند.

سرریزی پشته

- وقتی اشاره گر پشته به انتهای پشته می رسد پشته سرریز می شود. دلیل معمول سرریزی پشته فراخوانی مکرر یا تعداد زیاد متغیرهای محلی توابع بازگشتی است.
- اگر یک تابع بی نهایت بار خودش را صدا بزند در هر فراخوانی یک فریم پشته اضافه می شود و در یک نقطه پشته دیگر جا ندارد و سرریز می شود و خطای stack overflow رخ می دهد.

نمونه هائی از توابع بازگشتی

- مثال(C). تابع بازگشتي ضرب.

```
int Mul (int a , int b)
{
  if (b == 1)
    return a;
  else
    return a+Mul (a,b-1);
}
```

نمونه هائی از توابع بازگشتی

- مثال(Pascal). تابع بازگشتی فیبوناچی.

```
Function Fibonacci ( n: Integer ):Integer;
Begin
  If (n=1) or (n=2) Then Fib:=1
  Else Fibonacci:= Fibonacci (n-2)+ Fibonacci (n-1);
End;
```

با تشکر از توجه شما

By: Eng. M. Kamkarhaghighi