Capítulo 2

Indução e Recursão

"E de importância vital, distinguir a legibilidade e a facilidade de escrever programas. E importante ter a capacidade de escrever programas, mas é crucial que esses programas sejam faceia de ler e entender"

-Spender, Tremblay e Soreson-

Sumário:

2.1 - Indução Matemática
2.2- Recursividade
2.3 - Algoritmos definidos por Recorrência
2.4 - Estratégias Algorítmicas
2.5- Recomendações Bibliográficas
2.6- Exercícios

2.1- Indução Matemática

Suponhamos que temos uma escada infinita e queremos saber se podemos alcançar todos os degraus da escada. Mas, só podemos executar as seguintes acções:

1º- Podemos alcançar o primeiro degrau da escada;
2º- Se podemos alcançar um degrau da escada, então podemos alcançar o próximo degrau.

Com esse conhecimento, podemos alcançar todos os degraus da escada? Por (1) sabemos como alcançar o primeiro degrau. Como podemos alcançar o primeiro degrau, por (2) podemos alcançar o segundo degrau. Se aplicarmos novamente o segundo ponto podemos chegar ao terceiro degrau. Aplicando este processo de forma sucessiva, podemos alcançar o terceiro, quarto e assim por diante. Logo concluímos que podemos alcançar qualquer degrau da escada. A este processo de funcionamento dá-se o nome de **Indução Matemática** ou **indução Finita**.

A indução matemática é uma técnica muito importante para provar teoremas e para desenvolver algoritmos para computadores.

Neste capítulo veremos como está ferramenta pode ser utilizada para provar teoremas e nos próximos veremos a sua aplicação a computação, antes, porém faremos uma definição formal.

Seja n um número inteiro positivo e P(n) uma proposição qualquer. Provar que P(n) é verdadeiro para qualquer número inteiro positivo consiste em mostrar que:

Caso Base: P(1) é verdadeiro;

Passo Indutivo: Se P(n) por verdadeiro para um determinado n > 1 então P(n+1) também é verdadeiro

Para utilizar o Passo Indutivo, assumimos que existe um número positivo k, suficiente grande tal que P(k) é verdadeiro. Sobre essa hipótese, denominada por hipótese da indução vamos mostrar que P(k+1) também é verdadeira.

Para consolidar os conhecimentos, vejamos alguns exemplos muito simples:

Exemplo 1: mostre que se n for um número inteiro positivo então

$$1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$$

Resolução: Considere que P(n) é uma proposição que afirma que a soma dos primeiros números inteiros positivos é igual a n(n+1)/2.

Caso Base: de facto a proposição é verdadeira para n = 1, pois 1 = 1(1+1)/2.

Passo Indutivo: suponhamos pela hipótese da indução que a proposição é verdadeira para um número inteiro arbitrário k, ou seja, assumimos que

$$1 + 2 + 3 + ... + k = \frac{k(k+1)}{2} \quad (1)$$

Considerando que essa hipótese verdadeira, vamos mostrar que

$$1 + 2 + 3 + ... + k + (k+1) = \frac{(k+1)(k+2)}{2}$$

também é verdadeira.

Com efeito, se somarmos k+ 1 a ambos os membros da equação (1), obtemos

$$1 + 2 + 3 + ... + k + (k+1) = \frac{k(k+1)}{2} + (k+1)$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(K+1)(K+2)}{2}$$

Logo a proposição é verdadeira para qualquer K ≥ 1.

Exemplo 2: mostre que se n for um número inteiro positivo então

$$1 + 3 + 5 + ... + (2n-1) = n^2$$

Resolução: Considere que P(n) é uma proposição que afirma que a soma dos primeiros números inteiros positivos impares é igual a n^2.

<u>Caso Base</u>: de facto a proposição é verdadeira para n = 1, pois $1 = 1^2$.

<u>Passo Indutivo</u>: suponhamos pela hipótese da indução que a proposição é verdadeira para um número inteiro arbitrário k, ou seja, assumimos que

$$1 + 3 + 5 + ... + (2k-1) = k^2$$

Considerando que essa hipótese verdadeira, vamos mostrar que

$$1 + 3 + 5 + ...+ (2k -1)+ (2k+1) = (k+1)^2$$

também é verdadeira.

Com efeito,

$$\begin{aligned} 1 + 3 + 5 + ... (2k +1) &= [1 + 3 + 5 + ... (2k-1)] + (2k + 1) \\ &= k^2 + (2k + 1) \\ &= k^2 + 2k + 1 \\ &= (k + 1)^2 \end{aligned}$$

Logo a proposição é verdadeira para todo $k \geq 1$.

Exemplo 3: mostre que se n for um número inteiro não negativo então

$$1 + 2 + 2^2 + 2^3 + ... + 2^n = 2^{n+1} - 1$$

<u>Caso Base</u>: de facto a proposição é verdadeira para n = 1, pois $1 = 2^0$.

<u>Passo Indutivo</u>: suponhamos pela hipótese da indução que a proposição é verdadeira para um número inteiro arbitrário k, ou seja, assumimos que

$$1 + 2 + 2^2 + 2^3 + ... + 2^k = 2^{k+1} - 1$$

Considerando que essa hipótese verdadeira, vamos mostrar que:

$$1 + 2 + 2^2 + 2^3 + ... + 2^k + 2^{k+1} = 2^{k+2} - 1$$

Com efeito,

$$
\begin{aligned}
1 + 2 + 2^2 + 2^3 + \ldots\ldots + 2^k + 2^{k+1} &= (1 + 2 + 2^2 + 2^3 + \ldots\ldots + 2^k) + 2^{k+1} \\
&= 2^{k+1} - 1 + 2^{k+1} \\
&= 2^{k+1} + 2^{k+1} - 1 \\
&= 2 \times 2^{k+1} - 1 \\
&= 2^{k+2} - 1
\end{aligned}
$$

Logo a proposição é verdadeira para todo $k \geq 0$.

Exemplo 4: mostre que se n for um número inteiro positivo então

$$n < 2^n$$

<u>Caso Base</u>: de facto a proposição é verdadeira para n = 1, pois $1 < 2^1 = 2$.

<u>Passo Indutivo</u>: suponhamos pela hipótese da indução que a proposição é verdadeira para um número inteiro arbitrário k, ou seja, assumimos que

$$k < 2^k$$

Considerando essa hipótese verdadeira, vamos mostrar que

$$k + 1 < 2^{k+1}$$

Com efeito:

$$
\begin{aligned}
k + 1 &< 2^k + 1 \\
&< 2^k + 2^1 \\
&< 2^{k+1}
\end{aligned}
$$

Logo a proposição é verdadeira para todo $k \geq 0$.

2.2 - Recursão

Muitas vezes é difícil definir um objecto. Contudo essa definição fica mais simples se conseguirmos definir o objecto em função dele mesmo. A este tipo de definição dá-se o nome de **recursão**.

A recursão pode ser utilizada para definir inúmeros problemas no domínio da matemática e da computação.

Em termos gerais, uma fórmula baseada numa **Relação de Recorrência** ou **Recursão**, é descrita pelos seguintes passos:
<u>Caso Base</u>: Mostra de forma explicita os primeiros valores;
<u>Passo Indutivo</u>: expressa os restantes termos em função dos termos anteriores.

Exemplo 1: calcular a fórmula recorrente para a seguinte sequência de números inteiros

1, 2,3,4,5,.......

<u>Caso Base</u> :

$u_1 = 1$

<u>Passo indutivo</u>: vamos determinar uma fórmula que expresse qualquer termo da sequência em função dos termos anteriores.

$u_2 = 2$
$= 1 + 1$
$= u_1 + 1$

$u_3 = 3$
$= 2 + 1$
$= u_2 + 1$

Logo podemos concluir que:

$u_n = u_{n-1} + 1$

Então, a sequência anterior é descrita pela seguinte fórmula recorrente

$$u_n = \begin{cases} 1 & se\ n = 1 \\ u_{n-1} + 1 & se\ n > 1 \end{cases}$$

Exemplo 2: calcular a fórmula recorrente para a seguinte soma de números inteiros:

1+ 2 + 3 + 4 +

<u>Caso Base</u>:

$S_1 = 1$

<u>Passo indutivo</u>: vamos determinar uma fórmula que expresse qualquer valor em função dos valores anteriores.

$S_2 = 3$
$= 1 + 2$
$= S_1 + 2$

$S_3 = 6$
$= 3 + 3$
$= S_2 + 3$

Logo podemos concluir que:

$S_n = S_{n-1} + n$

Então, a soma anterior é descrita pela seguinte fórmula recorrente ou recursiva.

$$S_n = \begin{cases} 1 & se\ n = 1 \\ s_{n-1} + n & se\ n > 1 \end{cases}$$

Exemplo 3: calcular a fórmula recorrente para a seguinte soma de números inteiros.

$S = 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^n$

<u>Caso Base</u>:

$S_1 = 1$

<u>Passo indutivo</u>: vamos determinar uma fórmula que expresse qualquer valor em função dos valores anteriores.

$$\begin{aligned} S_2 &= 3 \\ &= 1 + 2 \\ &= S_1 + 1 + 1 \\ &= 2S_1 + 1 \end{aligned}$$

$$\begin{aligned} S_3 &= 7 \\ &= 3 + 4 \\ &= 3 + 3 + 1 \\ &= 2S_2 + 1 \end{aligned}$$

Então, a soma anterior pode ser descrita pela seguinte fórmula recorrente ou recursiva.

$$S_n = \begin{cases} 1 & se\ n = 1 \\ 2s_{n-1} + 1 & se\ n > 1 \end{cases}$$

2.3 – Algoritmos Definidos por Recorrência

Exemplo 1: desenvolva um programa para calcular o valor da seguinte fórmula recorrente.

$$S_n = \begin{cases} 2 & se\ n = 1 \\ 2s_{n-1} + 1 & se\ n > 1 \end{cases}$$

Podemos utilizar duas abordagens: Se pretendermos calcular S(8) por exemplo, começamos com S(1) que é igual a dois e, depois vamos calcular S(2), S(3) e assim por diante. Essa abordagem envolve uma iteração, ou seja, uma acção que se repete um número finito de vezes. A seguir, mostramos a função que implementa essa abordagem.

```
int S (int n)
{
   int i, valorCorrente;
    if ( n == 1)
       return 2;
   else
    {
       i = 2;
       valorCorrente = 2;
       while ( i <= n )
         {
           valorCorrente *= 2;
            i = i + 1;
         }
        return i;
     }
 }
```

A segunda abordagem consiste em utilizar as propriedades inerentes à fórmula recorrente, ou seja, as acções são tomadas por duas condições mutuamente exclusivas, que baseiam-se no princípio de indução.

<u>Caso Base</u>**:** garante a existência de uma condição de término, sem a qual a recursão seria infinita.

<u>Passo Recursivo</u>**:** garante a decomposição do problema em problemas mais simples.

Apresentamos a seguir uma função que implementa essa abordagem.

```
int S( int n)
{
    if ( n == 1 )
       return 2 ;                    /*  Caso Base  */
   else
       return 2*S(n-1) ;          /* Passo recursivo */
 }
```

Mas quando o comando **return** for acionado, a execução da função termina, e transfere da expressão que o sucede para o exterior. Logo, essa função pode ser escrita na notação compacta.

```
int S( int n)
{
   if ( n == 1 )   return 2 ;       /*  Caso Base  */
   return 2*s(n-1) ;               /* Passo Recursivo */
}
```

Para compreender o funcionamento desta função, veremos a seguir uma simulação descritiva. Antes, porém, daremos um conceito muito importante em computação.

Uma **pilha** é uma estrutura de dados, onde as operações de inserção (empilhar) e de remoção (desempilhar) são feitas numa única extremidade denominada por Topo. Essa estrutura goza da seguinte propriedade: O último elemento a entrar é o primeiro a sair.

Suponhamos sem perda da generalidade que n é igual a três. Como n não é igual a 1, caso base, a execução do programa é direccionada para a cláusula **else**. O cálculo de S(3) é suspenso e o valor do parâmetro é automaticamente inserido numa pilha em memória. Nesse instante a pilha contêm um único elemento.

3			
↑ topo			

Em seguida, a função volta a ser chamada para calcular o valor de S(2). Mais uma vez o fluxo de execução é direcionado para a clausula **else** e o cálculo de S(2) é suspenso. Como no caso anterior, o valor do parâmetro é automaticamente inserido numa pilha em memória. Nesse instante a pilha possui dois elementos.

3	2		
	↑ topo		

A função volta a ser chamada para calcular o valor de S(1) mas, como trata-se do caso base é retornado o valor dois. Como a chamada foi concluída, iniciamos o encerramento da última chamada suspensa. Esse encerramento consiste em remover o elemento que está no topo da pilha e multiplicar esse elemento por dois (relação de recorrência). O cálculo de S(2) é concluído e a pilha passa a ter um único elemento.

3			
↑ topo			

Como a pilha não está vazia então temos algumas chamadas suspensas. Iniciamos desse modo, o encerramento da penúltima chamada suspensa. Esse encerramento consiste em remover o elemento que está no topo da pilha e multiplicar esse elemento por 2 (relação de recorrência).

Agora a pilha está vazia, isso quer dizer que não temos mais chamadas suspensas, logo a função termina retornando o valor seis.

2.4- Estratégias Algorítmicas

Uma estratégia algorítmica é uma abordagem geral que é utilizada para desenvolver para problemas de diferentes áreas do conhecimento. É uma metodologia que aplica um conjunto de passos para abordar a solução de problemas. As estratégias algorítmicas que baseiam-se no princípio de recursão, podem ser classificados em função do número de chamadas que realizam e são chamadas der: Recursividade Linear (Decrementar e Conquistar), Recursividade em Árvore (Divisão e Conquista) e Recursividade com retrocesso (backtranking).

2.5- Recomendações Bibliográficas

Para o leitor aprofundar os temas abordados neste capítulo, recomendamos o livro:

Gersting J. L.;- *Mathematical Structures for Computer Science*, 5nd Edition, Freeman and Co, New York, 2003 .

Lehman E, Leighton T, Meyer A.;- *Mathematics for Computer Science*, 2010. (Disponível em http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/readings/).

2.6- Exercícios

2.6.1-Prove que $2^n > 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^0$, Para $n > 1$.

2.6.2- Prove que $1^3 + 2^2 + 2^3 + ... + 2^n = 2^{n+1} - 1$ para todo $n \geq 0$.

2.6.3- Prove que $1^3 + 3^3 + 5^3 + ... + (2n-1)^3 = 2n^4 - n^2$ para todo $n \geq 1$.

2.6.4- Prove que $1^3 + 2^2 + 2^3 + ... + 2^n = 2^{n+1} - 1$ para todo $n \geq 0$.

2.6.5- Prove que $2^n > n^2$, Para todo $n \geq 4$.

2.6.6-Determine a relação de recorrência que expressa a seguinte sequencia:

2, 4, 8, 16, …

2.6.7-Determine a relação de recorrência que expressa a seguinte série:

$1 + 1/2 + 1/3 + ... + 1/n$

2.6.8-Determine a relação de recorrência que expressa a seguinte sequência:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

2.6.9- Escreva os cinco primeiros termos da fórmula de recorrência.

$D(1) = 2$

$D(2) = 5$

$D(n) = (n-1)D(n-1) + (n-2)D(n-2)$ para todo $n > 2$

2.6.10- Escreva os sete primeiros termos da fórmula de recorrência.

$S(1) = 1$

$S(n) = S(n-1) + 1$ para todo $n \geq 2$