Procedimentos e funções

Recursividade

Prof. Fabio Takeshi Matsunaga

26 de agosto de 2019

Objetivos

Missão da aula de hoje

Entender o conceito da recursividade e a resolução de problemas com algoritmos. recursivos.

- Fazer os alunos aprenderem a desenvolver funções recursivas utilizando a linguagem de programação C;
- Fazer os alunos aprenderem técnicas para identificar os componentes de uma função recursiva - caso base e chamadas recursivas – em problemas recorrentes, como o cálculo do fatorial de um número inteiro;
- Incentivar os alunos a aplicarem algoritmos recursivos em outros problemas matemáticos.

Problemas

- Fatorial.
- Fibonacci.
- Maior valor de um vetor.
- Ordenação.

Vimos na última aula ...

Desenvolvimento de funções e subprogramas em C.

- Funções consistem em um conjunto de comandos agrupados em um bloco de código.
- Recebem um nome identificador a partir do qual é chamado pelo programa principal.
- Retornam um resultado para o programa principal.

Algoritmos iterativos

- Vimos também a implementação de funções para representar algoritmos iterativos.
- Algoritmos iterativos consistem na repetição de um conjunto de ações um determinado número de vezes até chegar ao resultado esperado.
- Nem todos os problemas são resolvidos de uma forma linear com uma única execução de uma operação.

Algoritmos iterativos

- Exemplo: algoritmo iterativo para resolver o problema do cálculo do fatorial de um número n.
- Para calcular um fatorial, é necessário repetir a operação de multiplicação n vezes

Algoritmo para calcular um fatorial

```
1  int Fatorial(int n) {
2          int f=1;
3          while (n>=1) {
4                  f = f*n;
5                  n--;
6          }
7          return f;
8  }
```

Recursividade

- Os mesmos problemas resolvidos com algoritmos iterativos podem ser resolvidos com algoritmos recursivos, que consiste em implementar uma função utilizando a recursividade.
- Recursividade é uma técnica que define uma solução de um problema em termos de uma ou mais versões menores do mesmo problema.
- Uma função que utiliza a recursividade consiste em uma função que chama a si mesma em algum comando.

Relações de recorrência

- A principal base matemática da recursividade é a relação de recorrência.
- Uma relação de recorrência é uma expressão que calcula o valor de uma função matemática em termos dos valores anteriores da mesma função.
- A expressão:
  - $F(n) = F(n-1) + n$
- consiste em uma recorrência que calcula o valor de $F(n)$ em relação ao valor de $F(n-1)$, considerando n o n-ésimo termo de uma sequência.

Relações de recorrência

- Considerando a expressão $F(n) = F(n-1) + n$ para $F(1) = 1$, os próximos termos da função podem ser calculados:
  - $F(2) = F(1) + 2 = 1 + 2 = 3$
  - $F(3) = F(2) + 3 = 3 + 3 = 6$
  - $F(4) = F(3) + 4 = 6 + 4 = 10$
  - $F(5) = F(4) + 5 = 10 + 5 = 15$

Estrutura de uma função recursiva

Considerando a relação de recorrência vista anteriormente, para aplicar a recursividade na programação, é necessário considerar os seguintes componentes de uma função recursiva:

- **Caso base**: caso simples e conhecido do problema.
- **Chamada recursiva**: chamada da própria função representando uma versão menor do problema que converge para o caso base.

Cálculo do fatorial de um número

- Sabemos que o fatorial de um valor se dá pela fórmula $F(n) = n * (n-1) * (n-2) * ... * 1$, considerando n um número inteiro e positivo.
- Exemplo: $4! = 4 * 3 * 2 * 1 = 24$
- Vamos implementar a sua solução recursiva.

Cálculo do fatorial de um número

Identificando os componentes da chamada recursiva do cálculo do fatorial.

- **Caso base**: sabe-se o caso mais básico do fatorial é $1! = 1$.
- **Chamada recursiva**: para calcular o fatorial de um número n, calcula-se o fatorial de $n - 1$ que multiplica com o valor de n.

Algoritmo recursivo para o fatorial

Para $n = 4$ e $n - 1 = 3$ como 4! = 4*3*2*1, pode-se dizer que:

- 4! = 4*(3!)
- 4! = 4*3*2*1
- 4! = 4*3*(2!)
- 4! = 4*3*2*1
- 4! = 4*3*2*(1!) (1!=1 é o caso base)

Logo, pode-se generalizar que o cálculo do fatorial de um número inteiro e positivo é dado pelo número multiplicado pelo fatorial do número anterior, a partir do qual uma nova chamada do cálculo de fatorial é realizada.

Algoritmo recursivo para o fatorial

Logo, a relação de recorrência para o cálculo do fatorial $F(n)$ de um número é dada por:

$$\begin{cases} F(1) = 1 \\ F(n) = n * F(n-1) \end{cases}$$

Analisando a pilha de recursão

Aplicando o exemplo para $n = 5$:

1. $F(5) = 5 * F(4)$

2. $F(5) = 5 * 4 * F(3)$

3. $F(5) = 5 * 4 * 3 * F(2)$

4. $F(5) = 5 * 4 * 3 * 2 * F(1)$ → Convergiu para o caso base.

5. $F(5) = 5 * 4 * 3 * 2 * 1$

Analisando a pilha de recursão

Executando a pilha de recursão até chegar ao problema original ($F(5)$):

- $F(5) = 5 * 4 * 3 * \boxed{2 * 1}$
- $F(5) = 5 * 4 * \boxed{3 * \underline{2}}$
- $F(5) = 5 * \boxed{4 * \underline{6}}$
- $F(5) = \boxed{5 * \underline{24}}$
- $F(5) = \underline{120}$

Implementando em linguagem C

```
1  int Fatorial(int n) {
2      if (n==1) return 1;
3      else return n*Fatorial(n-1);
4  }
```

Explicação da função recursiva

- Quando uma função chama a si mesma, novos parâmetros e variáveis locais são alocados na pilha de recursão.
- Cada chamada recursiva em Fatorial(n-1) invoca uma nova chamada da função, com um parâmetro diferente.
- Como cada chamada recursiva diminui o valor de n em 1 unidade, a tendência é em algum momento do algoritmo convergir para $n = 1$, que é o caso base.
- Depois que a chamada recursiva convergir, as operações da pilha de recursão são executadas da direita para esquerda, para produzir o resultado final.

Conclusões

- Uma função recursiva cria uma versão mais clara e simplificada de um algoritmo com relação à mesma versão iterativa.
- A aplicação de métodos recursivos contribui para a construção de códigos-fontes elegantes.
- Apesar das vantagens mencionadas a implementação de funções recursivas requer uma base matemática na área de recorrências e seu uso deve ser cuidadoso por consumir mais memória.

Atividades

1. Implemente uma função recursiva em C para a relação de recorrência $F(n) = F(n-1) + n$ considerando o caso base $F(1) = 1$. Em seguida, analise a lógica da sequência numérica gerada pelos resultados.
2. Implemente uma função recursiva em C que forneça como entrada dois números: x e n. Em seguida, a função deve calcular e retornar o valor de x^n.
3. Implemente uma função recursiva em C para calcular o valor n da sequência de Fibonacci $= \{1,1,2,3,5,8,13...\}$, que consiste na soma dos dois elementos anteriores. A relação de recorrência pode ser definida por $F(n) = F(n-1) + F(n-2)$ e o caso base $F(1) = F(2) = 1$.
4. Implemente uma função recursiva em C que receba como entrada um vetor de inteiros e seu tamanho e em seguida calcule e retorne a soma de todos os elementos do vetor.

Bibliografia recomendada

- SCHILDT, Herbert; MAYER, Roberto Carlos. **C completo e total**. 2006.
- MIZRAHI, Victorine Viviane. **Treinamento em linguagem C**. McGraw-Hill, 1990.

OBRIGADO!

“Quem só acredita no visível tem um mundo muito pequeno.”
(Caio Fernando Abreu)

Contato

- **Fabio Takeshi Matsunaga**
- E-mail: **matsunaga@utfpr.edu.br**
- Permanência: **sala K003**;