USP-ICMC-BMACC/BCDados

SCC0223 - Estrutura de Dados I

2023

# Resolução da Lista de Exercícios 1

1. ★ Mostre, através de teste de mesa, o resultado das seguintes funções:

   i.
```
int f1(int n)
{
  if (n == 0)
    return (1);
  else
    return(n * f1(n-1));
}
```
   Considere as entradas:

   (1) $f1(0)$;
   (2) $f1(1)$;
   (3) $f1(5)$;

   *Resolução*:
```
int cont=0;
int f1(int n)
{
  cont++; // if
  if (n == 0)
  {
    cont++; // return
    return (1);
  }
  else
  {
    cont++; // return
    return(n * f1(n-1));
  }
}
```
   f1(0): 1
   CONTADOR para f1(0): 2
   f1(1): 1
   CONTADOR para f1(1): 4
   f1(5): 120
   CONTADOR para f1(5): 12

 Veja a próxima página...

ii.
```
int f2(int n)
{
  if (n == 0)
    return (1);
  if (n == 1)
    return (1);
  else
    return(f2(n-1)+ 2 * f2(n-2));
}
```

Considere as entradas:

(1) $f2(0)$;
(2) $f2(1)$;
(3) $f2(5)$;

*Resolução*:

```
int cont=0;
int f2(int n)
{
  cont++; // if
  if (n == 0)
  {
    cont++; // return
    return (1);
  }
  cont++; // if
  if (n == 1)
  {
    cont++; // return
    return (1);
  }
  else
  {
    cont++; // return
    return(f2(n-1)+ 2 * f2(n-2));
  }
}
```

f2(0): 1
CONTADOR para f2(0): 2
f2(1): 1
CONTADOR para f2(1): 3
f2(5): 21
CONTADOR para f2(5): 42

iii.
```
int f3(int n)
{
  if (n == 0)
    printf(``Zero '');
  else
  {
    printf(``%d '',n);
    printf(``%d '',n);
    f3(n-1);
  }
}
```

Considere as entradas:

(1) $f3(0)$;
(2) $f3(1)$;
(3) $f3(5)$;

*Resolução*:

```
int cont=0;
int f3(int n)
{
  cont++; // if
  if (n == 0)
  {
    cont++; // printf
    printf(``Zero\n'');
  }
  else
  {
    cont+=3; // printfs e chamada
    printf(``%d \n'',n);
    printf(``%d \n'',n);
    f3(n-1);
  }
}
```

Zero
CONTADOR para f3(0): 2
1
1
Zero
CONTADOR para f3(1): 6
5

5
4
4
3
3
2
2
1
1
Zero
CONTADOR para f3(5): 22

2. Desenvolva algoritmos recursivos para os seguintes problemas:

   i. ★ Impressão de um número natural em base binária.

   *Resolução*:

```
void bin(int n)
{
  if (n > 1)
    bin(n/2);
  printf(``%d'',n%2);
}
```

   ii. ★ Multiplicação de dois números naturais, através de somas sucessivas (Ex.: $6*4 = 4+4+4+4+4+4$).

   *Resolução*:

```
int multsoma(int a, int b)
{
  if (a > 0)
    return b+multsoma(a-1,b);
  else
    return 0;
}
```

   iii. ★ Soma de dois números naturais, através de incrementos sucessivos (Ex.: $3+2 = ++(++3)$).

   *Resolução*:

```
int somaincr(int a, int b)
{
  if (b > 0)
    return somaincr(++a,b-1);
```

```
  else
    return a;
}
```

iv. ★ Multiplicação de dois números naturais, através de incrementos sucessivos.

*Resolução*:

```
int somaincr2(int x, int y)
{
    if (y == 0)
        return x;
    else
        return somaincr2(++x,--y);
}

int prodincr(int x, int y)
{
    if (y == 0 || x == 0)
        return 0;
    else
        return somaincr2(x, prodincr(x,--y));
}
```

v. ★ Cálculo de $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{N}$.

*Resolução*:

```
float somafrac(float n)
{
  if (n > 0)
    return 1/n+somafrac(n-1);
  else
    return 0;
}
```

vi. ★ Cálculo de $\frac{2}{4} + \frac{5}{5} + \frac{10}{6} + \frac{17}{7} + \frac{26}{8} + ... + \frac{(n^2+1)}{(n+3)}$.

*Resolução*:

```
float somafrac2(float n)
{
  if (n > 0)
    return (n*n+1)/(n+3)+somafrac2(n-1);
  else
```

```
    return 0;
}
```

vii. ★ Inversão de uma string.

*Resolução*:

```
void inverte(char *ch, int inicio, int fim)
{
   char c;
   if (inicio >= fim)
      return;
   c = *(ch+inicio);
   *(ch+inicio) = *(ch+fim);
   *(ch+fim) = c;
   inverte(ch, ++inicio, --fim);
}
```

viii. Gerador da sequência dada por:

- $F(1) = 1$
- $F(2) = 2$
- $F(n) = 2 * F(n-1) + 3 * F(n-2)$.

ix. Gerador da sequência: $1, 2, 5, 12, 29, 68, 165, ....$

x. Gerador da sequência: $0, 1, 1, 2, 3, 7, 16, 65, 321, ....$

xi. ★ Gerador de Sequência de Ackerman:

- $A(m, n) = n + 1$, se $m = 0$
- $A(m, n) = A(m-1, 1)$, se $m \neq 0$ e $n = 0$
- $A(m, n) = A(m-1, A(m, n-1))$, se $m \neq 0$ e $n \neq 0$.

*Resolução*:

```
int ackermann(unsigned m, unsigned n)
{
  if (m == 0)
    return n + 1;
  else
    if (n == 0)
      return ackermann(m-1,1);
    else
      return ackermann(m-1,ackermann(m,n-1));
}
```

xii. A partir de um vetor de números inteiros, calcule a soma e o produto dos elementos do vetor.

xiii. Gerador de máximo divisor comum (mdc):
- $mdc(x, y) = y$, se $x \geq y$ e $x \; mod \; y = 0$
- $mdc(x, y) = mdc(y, x)$, se $x < y$
- $mdc(x, y) = mdc(y, x \; mod \; y)$, caso contrário.

xiv. Verifique se uma palavra é palíndromo (Ex. *aba*, *abcba*, *xyzzyx*).

xv. Dado um número $n$, gere todas as possíveis combinações com as $n$ primeiras letras do alfabeto. Ex.: $n = 3$. Resposta: ABC, ACB, BAC, BCA, CAB, CBA.

xvi. Gere todas as possíveis combinações para um jogo da MegaSena com 6 dezenas.

3. ★ Verifique o que as funções dos algoritmos abaixo imprimem e retornam:

i.
```
func (int n)
{
  if (n == 0)
    printf(``fim'');
  else
  {
    printf(n);
    func(n-1);
  }
}
```

ii.
```
func (int n)
{
  if (n == 0)
    printf(``fim'');
  else
  {
    func(n-1);
    printf(n);
  }
}
```

iii.
```
func (int n)
{
  if (n == 0)
    printf(``fim'');
  else
  {
    printf(n);
    func(n-1);
    printf(n);
  }
}
```

iv.
```
func (int n)
{
  if (n == 0)
    printf(``fim'');
  else
  {
    func(n-1);
    printf(n);
    func(n-1);
  }
}
```

*Resolução*:

Para n = 3:

func1:
3
2
1
fim

func2:
fim
1
2
3

func3:
3
2
1
fim
1
2
3

func4:
fim
1
fim

2
fim
1
fim
3
fim
1
fim
2
fim
1
fim

4. Compare e explique o funcionamento dos algoritmos do exercício anterior.

5. Escreva um programa recursivo em C para classificar um vetor $a$ como segue:

   - Seja $k$ o índice do elemento do meio do vetor.
   - Classifique os elementos até, e inclusive, $a[k]$.
   - Classifique os elementos depois de $a[k]$.
   - Combine os dois subvetores num único vetor classificado.

   Este método é chamado de **classificação mesclada**.

6. ★ Determine o que a seguinte função recursiva em C calcula. Escreva uma função iterativa para atingir o mesmo objetivo.

```
func (int n)
{
  if (n == 0)
    return(0);
  return(n + func(n-1));
}
```

*Resolução*:

```
int func_i (int n)
{
  int i, soma=0;
  for (i = 0; i <= n; i++)
    soma+=i;
```

```
    return soma;
}
```

7. Defina uma **sequência de Fibonacci generalizada**, de $f0$ a $f1$ como sequência $fibg(f0, f1, 0)$, $fibg(f0, f1, 1)$, $fibg(f0, f1, 2)$, ..., onde:

   - $fibg(f0, f1, 0) = f0$
   - $fibg(f0, f1, 1) = f1$
   - $fibg(f0, f1, n) = fibg(f0, f1, n-1) + fibg(f0, f1, n-2)$, se $n > 1$.

   Escreva uma função recursiva em C para calcular $fibg(f0, f1, n)$. Descubra um método iterativo para calcular essa função.

# Referências

[1] Nakamiti, Gilberto, *Listas de Exercícios de Estruturas de Dados II*," Engenharia de Computação. PUC-Campinas, 2007.

[2] Tenenbaum, A. M., Langsam, Y., Augestein, M. J., *Estruturas de Dados Usando C*. Makron Books, 1995.