



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

Disciplina Algoritmos e Estruturas de Dados I	Curso Ciência da Computação	Turno Manhã	Período 1º
Professor Felipe Cunha (felipe@pucminas.br)			

Lista de Exercícios 04 - Correção

1. Fazer uma função *int par(int n)* que recebe um número inteiro *n* e retorna o *n*-ésimo termo da sequência 2, 4, 6, 8, 10, 12.... Além disso, você deve fazer um procedimento *exercicio01()* para ler o valor de *n* e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função que retorna o n-ésimo termo da sequência de pares
4 int par(int n) {
5     return 2 * n;
6 }
7
8 // Procedimento solicitado
9 void exercicio01() {
10     int n, resultado;
11
12     printf("Digite um numero inteiro n: ");
13     scanf("%d", &n);
14
15     resultado = par(n);
16
17     printf("O %d-esimo termo da sequencia de pares eh: %d\n", n,
18         resultado);
19 }
20 // Função principal apenas para rodar o exercício
21 int main() {
22     exercicio01();
23     return 0;
24 }
```

2. Fazer uma função *void mostrarParesEmOrdemDecrescente(int n)* que recebe um número inteiro *n* e mostra na tela (em ordem decrescente) todos os valores menores do que *n* para a sequência do exercício anterior. A sua função *mostrarParesEmOrdemDecrescente* deve utilizar a função *par* desenvolvida na questão anterior. Além disso, você deve fazer um procedimento *exercicio02()* para ler o valor de *n* e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função auxiliar (já deveria existir da questão anterior)
4 int par(int x) {
5     return x % 2 == 0;
6 }
7
8 // Função para mostrar pares em ordem decrescente menores que n
9 void mostrarParesEmOrdemDecrescente(int n) {
10     printf("Numeros pares menores que %d em ordem decrescente:\n", n);
11
12     for (int i = n - 1; i >= 0; i--) {
13         if (par(i)) {
14             printf("%d ", i);
15         }
16     }
17     printf("\n");
18 }
19
20 // Procedimento exercicio02
21 void exercicio02() {
22     int n;
23     printf("Digite um numero inteiro: ");
24     scanf("%d", &n);
25
26     mostrarParesEmOrdemDecrescente(n);
27 }
28
29 // Função principal só para rodar o exercício
30 int main() {
31     exercicio02();
32     return 0;
33 }
```

3. Fazer uma função *double umSobreImpar(int n)* que recebe um número inteiro *n* e retorna o *n*-ésimo termo da sequência $\frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}, \frac{1}{11}$ Além disso, você deve fazer um procedimento *exercicio03()* para ler o valor de *n* e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7, ...
4 double umSobreImpar(int n) {
5     return 1.0 / (2 * n + 1);
6 }
7
8 // Procedimento solicitado
9 void exercicio03() {
10     int n;
11     double resultado;
12
13     printf("Digite um numero inteiro n: ");
14     scanf("%d", &n);
15
16     resultado = umSobreImpar(n);
17
18     printf("O %d-esimo termo da sequencia 1/3, 1/5, 1/7... eh: %.6f\n",
19           n, resultado);
20 }
21
22 // Função principal apenas para rodar o exercício
23 int main() {
24     exercicio03();
25     return 0;
26 }
```

4. Fazer uma função *double somaUmSobreImpar(int n)* que recebe um número inteiro *n* e retorna o valor do somatório dos *n* primeiros termos da sequência anterior. A sua função *somaUmSobreImpar* deve utilizar a função *umSobreImpar* desenvolvida na questão anterior. Além disso, você deve fazer um procedimento *exercicio04()* para ler o valor de *n* e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7, ...
4 double umSobreImpar(int n) {
5     return 1.0 / (2 * n + 1);
6 }
7
8 // Função que retorna o somatório dos n primeiros termos da sequência
9 double somaUmSobreImpar(int n) {
10     double soma = 0.0;
11     for (int i = 1; i <= n; i++) {
12         soma += umSobreImpar(i);
13     }
14     return soma;
15 }
16
17 // Procedimento solicitado
18 void exercicio04() {
19     int n;
20     double resultado;
21
22     printf("Digite um numero inteiro n: ");
23     scanf("%d", &n);
24
25     resultado = somaUmSobreImpar(n);
26
27     printf("O somatorio dos %d primeiros termos da sequencia eh:
28         %.6f\n", n, resultado);
29 }
30
31 // Função principal apenas para rodar o exercício
32 int main() {
33     exercicio04();
34     return 0;
35 }
```

5. Fazer:

- (a) Uma função *double* *parSobreImpar(int n)* que recebe um número inteiro *n* e retorna o *n*-ésimo termo da sequência $\frac{2}{3}, \frac{4}{5}, \frac{6}{7}, \frac{8}{9}, \frac{10}{11}, \dots$. A sua função deve utilizar as funções *par* e *umSobreImpar* desenvolvidas anteriormente.
- (b) Uma função *double* *somaParSobreImpar(int n)* que recebe um número inteiro *n* e retorna o valor do somatório dos *n* primeiros termos da sequência anterior. A sua função *somaParSobreImpar* deve utilizar a função *parSobreImpar*.
- (c) Um procedimento *exercicio05()* para ler o valor de *n* e chamar as funções desenvolvidas nesta questão.

```
1 #include <stdio.h>
2
3 int par(int n) {
4     return 2 * n;
5 }
6
7 double umSobreImpar(int n) {
8     return 1.0 / (2 * n + 1);
9 }
10
11 // (a) Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
12 double parSobreImpar(int n) {
13     return par(n) * umSobreImpar(n);
14 }
15
16 // (b) Função que retorna o somatório dos n primeiros termos da
17 // sequência
18 double somaParSobreImpar(int n) {
19     double soma = 0.0;
20     for (int i = 1; i <= n; i++) {
21         soma += parSobreImpar(i);
22     }
23     return soma;
24 }
25
26 // (c) Procedimento exercicio05
27 void exercicio05() {
28     int n;
29     double termo, soma;
30
31     printf("Digite um numero inteiro n: ");
32     scanf("%d", &n);
33
34     termo = parSobreImpar(n);
35     soma = somaParSobreImpar(n);
36
37     printf("O %d-esimo termo da sequencia (2/3, 4/5, ...) eh: %.6f\n",
38           n, termo);
39     printf("O somatorio dos %d primeiros termos da sequencia eh:
40           %.6f\n", n, soma);
41 }
42
43 // Função principal apenas para rodar o exercício
44 int main() {
45     exercicio05();
46     return 0;
47 }
```

6. Fazer a função que recebe um número inteiro n , um número real x e retorna o n -ésimo termo da sequência abaixo. Utilize a função desenvolvida na letra a da questão anterior. Além disso, você deve fazer um procedimento `exercicio06()` para ler os valores de n e de x , e chamar a função desenvolvida nesta questão.

$$\frac{2x}{3}, \frac{4x^2}{5}, \frac{6x^3}{7}, \frac{8x^4}{9}, \frac{10x^5}{11} \dots$$

```
1 #include <stdio.h>
2 #include <math.h>
3
4 // Função que retorna o n-ésimo termo da sequência de pares (2, 4, 6,
5 // 8...)
6 int par(int n) {
7     return 2 * n;
8 }
9
10 // Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7...
11 double umSobreImpar(int n) {
12     return 1.0 / (2 * n + 1);
13 }
14
15 // Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
16 double parSobreImpar(int n) {
17     return par(n) * umSobreImpar(n);
18 }
19
20 // Função que retorna o n-ésimo termo da sequência com x
21 // (2x/3, 4x^2/5, 6x^3/7, ...)
22 double termoComX(int n, double x) {
23     return parSobreImpar(n) * pow(x, n);
24 }
25
26 // Procedimento exercicio06
27 void exercicio06() {
28     int n;
29     double x, resultado;
30
31     printf("Digite um numero inteiro n: ");
32     scanf("%d", &n);
33
34     printf("Digite um numero real x: ");
35     scanf("%lf", &x);
36
37     resultado = termoComX(n, x);
38
39     printf("O %d-esimo termo da sequencia com x eh: %.6f\n", n,
40         resultado);
41 }
42
43 // Função principal apenas para rodar o exercício
44 int main() {
45     exercicio06();
46     return 0;
47 }
```

7. Fazer uma função que recebe um número inteiro n , um real x e retorna o produto dos n primeiros termos da sequência acima. Utilize a função desenvolvida na questão anterior. Além disso, você deve fazer um procedimento `exercicio07()` para ler os valores de n e de x , e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int par(int n) {
5     return 2 * n;
6 }
7
8 double umSobreImpar(int n) {
9     return 1.0 / (2 * n + 1);
10 }
11
12 // Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
13 double parSobreImpar(int n) {
14     return par(n) * umSobreImpar(n);
15 }
16
17 // Função que retorna o n-ésimo termo da sequência com x
18 // (2x/3, 4x^2/5, 6x^3/7, ...)
19 double termoComX(int n, double x) {
20     return parSobreImpar(n) * pow(x, n);
21 }
22
23 // Função que retorna o produto dos n primeiros termos da sequência
24 double produtoTermos(int n, double x) {
25     double produto = 1.0;
26     for (int i = 1; i <= n; i++) {
27         produto *= termoComX(i, x);
28     }
29     return produto;
30 }
31
32 // Procedimento exercicio07
33 void exercicio07() {
34     int n;
35     double x, resultado;
36
37     printf("Digite um numero inteiro n: ");
38     scanf("%d", &n);
39
40     printf("Digite um numero real x: ");
41     scanf("%lf", &x);
42
43     resultado = produtoTermos(n, x);
44
45     printf("O produto dos %d primeiros termos da sequencia com x eh:
46         %.6f\n", n, resultado);
47 }
48
49 // Função principal apenas para rodar o exercício
50 int main() {
51     exercicio07();
52     return 0;
53 }
```

8. Fazer uma função que recebe um número inteiro n e retorna o seu fatorial¹. Além disso, você deve fazer um procedimento `exercicio08()` para ler o valor de n e chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função que retorna o fatorial de n
4 double fatorial(int n) {
5     double resultado = 1.0;
6     for (int i = 1; i <= n; i++) {
7         resultado *= i;
8     }
9     return resultado;
10 }
11
12 // Procedimento exercicio08
13 void exercicio08() {
14     int n;
15     double resultado;
16
17     printf("Digite um numero inteiro n: ");
18     scanf("%d", &n);
19
20     resultado = fatorial(n);
21
22     printf("O fatorial de %d eh: %.0f\n", n, resultado);
23 }
24
25 // Função principal apenas para rodar o exercício
26 int main() {
27     exercicio08();
28     return 0;
29 }
```

¹DICA: Apesar do fatorial ser uma função inteira, é recomendável retornar um número do tipo `double`.

9. Fazer uma função que recebe um número inteiro n , um número real x e retorna o n -ésimo termo da sequência abaixo. Utilize as funções desenvolvidas anteriormente. Além disso, você deve fazer um procedimento `exercicio09()` para ler o valor de n e chamar a função desenvolvida nesta questão.

$$\frac{2x}{3!}, \frac{4x^2}{5!}, \frac{6x^3}{7!}, \frac{8x^4}{9!}, \frac{10x^5}{11!} \dots$$

Observação: A partir deste ponto, sempre que possível, utilize funções desenvolvidas anteriormente.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 /* Reuso das funções anteriores */
5 int par(int n) {
6     return 2 * n;           // 2, 4, 6, ...
7 }
8
9 double fatorial(int n) {    // retorna n! como double
10     if (n < 0) return 0.0;  // simples proteção
11     double r = 1.0;
12     for (int i = 1; i <= n; i++) r *= i;
13     return r;
14 }
15
16 /* n-ésimo termo: (2n * x^n) / (2n + 1)! */
17 double termoFatorialComX(int n, double x) {
18     if (n < 1) return 0.0;  // sequência indexada a partir de n=1
19     int numPar = par(n);    // 2n
20     int fatIndice = 2 * n + 1; // (2n+1)!
21     return (numPar * pow(x, n)) / fatorial(fatIndice);
22 }
23
24 /* exercicio09: lê n e x e mostra o n-ésimo termo */
25 void exercicio09() {
26     int n;
27     double x;
28
29     printf("Digite um numero inteiro n: ");
30     scanf("%d", &n);
31
32     printf("Digite um numero real x: ");
33     scanf("%lf", &x);
34
35     double termo = termoFatorialComX(n, x);
36     printf("O %d-esimo termo da sequencia eh: %.10f\n", n, termo);
37 }
38
39 /* main apenas para executar o exercicio */
40 int main() {
41     exercicio09();
42     return 0;
43 }
```

10. Fazer uma função que recebe um número inteiro n , um número real x e retorna o somatório dos n primeiros termos da sequência mostrada na questão anterior. Além disso, você deve fazer um procedimento `exercicioXX()` para chamar a função desenvolvida nesta questão.

```
1  #include <stdio.h>
2  #include <math.h>
3
4  /* Reuso de funções anteriores */
5  int par(int n) {
6      return 2 * n;    // 2, 4, 6, ...
7  }
8
9  double fatorial(int n) {
10     if (n < 0) return 0.0;
11     double r = 1.0;
12     for (int i = 1; i <= n; i++) r *= i;
13     return r;
14 }
15
16 /* n-ésimo termo: (2n * x^n) / (2n+1)! */
17 double termoFatorialComX(int n, double x) {
18     if (n < 1) return 0.0;
19     return (par(n) * pow(x, n)) / fatorial(2 * n + 1);
20 }
21
22 /* Somatório dos n primeiros termos */
23 double somaTermosFatorialComX(int n, double x) {
24     double soma = 0.0;
25     for (int i = 1; i <= n; i++) {
26         soma += termoFatorialComX(i, x);
27     }
28     return soma;
29 }
30
31 /* Procedimento solicitado */
32 void exercicioXX() {
33     int n;
34     double x;
35
36     printf("Digite um numero inteiro n: ");
37     scanf("%d", &n);
38
39     printf("Digite um numero real x: ");
40     scanf("%lf", &x);
41
42     double resultado = somaTermosFatorialComX(n, x);
43
44     printf("O somatorio dos %d primeiros termos da sequencia eh:
45         %.10f\n", n, resultado);
46 }
47
48 /* main apenas para teste */
49 int main() {
50     exercicioXX();
51     return 0;
52 }
```

11. Refazer a função da questão anterior, omitindo todos os termos cujos valores no denominador sejam múltiplos de 2. Além disso, você deve fazer um procedimento `exercicioXX()` para chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int par(int n) {
5     return 2 * n;    // 2, 4, 6, ...
6 }
7
8 double fatorial(int n) {
9     if (n < 0) return 0.0;
10    double r = 1.0;
11    for (int i = 1; i <= n; i++) r *= i;
12    return r;
13 }
14
15 /* n-ésimo termo: (2n * x^n) / (2n+1)! */
16 double termoFatorialComX(int n, double x) {
17     if (n < 1) return 0.0;
18     return (par(n) * pow(x, n)) / fatorial(2 * n + 1);
19 }
20
21 /* Somatório omitindo termos cujo denominador seja múltiplo de 2 */
22 double somaTermosOmitindoDenominadorPar(int n, double x) {
23     double soma = 0.0;
24     for (int i = 1; i <= n; i++) {
25         int denomIndex = 2 * i + 1;
26         double denom = fatorial(denomIndex);
27
28         if (fmod(denom, 2.0) != 0.0) { // só soma se não for múltiplo
29             soma += termoFatorialComX(i, x);
30         }
31     }
32     return soma;
33 }
34
35 void exercicioXX() {
36     int n;
37     double x;
38
39     printf("Digite um numero inteiro n: ");
40     scanf("%d", &n);
41
42     printf("Digite um numero real x: ");
43     scanf("%lf", &x);
44
45     double resultado = somaTermosOmitindoDenominadorPar(n, x);
46
47     printf("O somatorio (omitindo termos com denominador multiplo de 2)
48         eh: %.10f\n", resultado);
49 }
50
51 int main() {
52     exercicioXX();
53     return 0;
54 }
```

12. Fazer uma função que recebe um número inteiro n e imprima os n primeiros múltiplos de 5. Além disso, você deve fazer um procedimento `exercicioXX()` para chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 /* Função que imprime os n primeiros múltiplos de 5 */
4 void multiplosDe5(int n) {
5     printf("Os %d primeiros multiplos de 5 sao:\n", n);
6     for (int i = 1; i <= n; i++) {
7         printf("%d ", 5 * i);
8     }
9     printf("\n");
10 }
11
12 /* Procedimento solicitado */
13 void exercicioXX() {
14     int n;
15
16     printf("Digite um numero inteiro n: ");
17     scanf("%d", &n);
18
19     multiplosDe5(n);
20 }
21
22 /* Função principal para rodar o exercício */
23 int main() {
24     exercicioXX();
25     return 0;
26 }
```

13. Fazer uma função que recebe um número inteiro n e retorna o n -ésimo termo da sequência de Fibonacci. Além disso, você deve fazer um procedimento `exercicioXX()` para chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 /* Função que retorna o n-ésimo termo da sequência de Fibonacci */
4 int fibonacci(int n) {
5     if (n == 0) return 0;
6     if (n == 1) return 1;
7     return fibonacci(n - 1) + fibonacci(n - 2);
8 }
9
10 /* Procedimento solicitado */
11 void exercicioXX() {
12     int n, resultado;
13
14     printf("Digite um numero inteiro n: ");
15     scanf("%d", &n);
16
17     resultado = fibonacci(n);
18
19     printf("O %d-esimo termo da sequencia de Fibonacci eh: %d\n", n,
20         resultado);
21 }
22
23 /* Função principal apenas para rodar o exercício */
24 int main() {
25     exercicioXX();
26     return 0;
27 }
```

14. Fazer uma função que recebe um número inteiro n e retorna o maior elemento da sequência de Fibonacci que seja menor que n. Além disso, você deve fazer um procedimento `exercicioXX()` para chamar a função desenvolvida nesta questão.

```
1 #include <stdio.h>
2
3 // Função que retorna o maior elemento de Fibonacci menor que n
4 int maiorFibonacciMenorQueN(int n) {
5     int a = 0, b = 1, prox = 0;
6
7     // Caso n seja menor ou igual a 1, não existe Fibonacci menor que n
8     if (n <= 1) {
9         return -1;
10    }
11
12    while (b < n) {
13        prox = a + b;
14        a = b;
15        b = prox;
16    }
17
18    return a; // a será o maior Fibonacci menor que n
19 }
20
21 // Procedimento solicitado
22 void exercicioXX() {
23     int n, resultado;
24
25     printf("Digite um numero inteiro: ");
26     scanf("%d", &n);
27
28     resultado = maiorFibonacciMenorQueN(n);
29
30     if (resultado == -1) {
31         printf("Nao existe numero de Fibonacci menor que %d.\n", n);
32     } else {
33         printf("O maior numero de Fibonacci menor que %d eh: %d\n", n,
34             resultado);
35     }
36 }
37
38 int main() {
39     exercicioXX();
40     return 0;
41 }
```