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

Disciplina	Curso	Turno	Período
Algorítmos e Estruturas de Dados I	Ciência da Computação	Manhã	1°

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.

```c
#include <stdio.h>

// Função que retorna o n-ésimo termo da sequência de pares
int par(int n) {
    return 2 * n;
}

// Procedimento solicitado
void exercicio01() {
    int n, resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    resultado = par(n);

    printf("O %d-esimo termo da sequencia de pares eh: %d\n", n,
       resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio01();
    return 0;
}
```

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.

```c
#include <stdio.h>

// Função auxiliar (já deveria existir da questão anterior)
int par(int x) {
    return x % 2 == 0;
}

// Função para mostrar pares em ordem decrescente menores que n
void mostrarParesEmOrdemDecrescente(int n) {
    printf("Numeros pares menores que %d em ordem decrescente:\n", n);

    for (int i = n - 1; i >= 0; i--) {
        if (par(i)) {
            printf("%d ", i);
        }
    }
    printf("\n");
}

// Procedimento exercicio02
void exercicio02() {
    int n;
    printf("Digite um numero inteiro: ");
    scanf("%d", &n);

    mostrarParesEmOrdemDecrescente(n);
}

// Função principal só para rodar o exercício
int main() {
    exercicio02();
    return 0;
}
```

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.

```c
#include <stdio.h>

// Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7, ...
double umSobreImpar(int n) {
    return 1.0 / (2 * n + 1);
}

// Procedimento solicitado
void exercicio03() {
    int n;
    double resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    resultado = umSobreImpar(n);

    printf("O %d-esimo termo da sequencia 1/3, 1/5, 1/7... eh: %.6f\n",
       n, resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio03();
    return 0;
}
```

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.

```c
#include <stdio.h>

// Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7, ...
double umSobreImpar(int n) {
    return 1.0 / (2 * n + 1);
}

// Função que retorna o somatório dos n primeiros termos da sequência
double somaUmSobreImpar(int n) {
    double soma = 0.0;
    for (int i = 1; i <= n; i++) {
        soma += umSobreImpar(i);
    }
    return soma;
}

// Procedimento solicitado
void exercicio04() {
    int n;
    double resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    resultado = somaUmSobreImpar(n);

    printf("O somatorio dos %d primeiros termos da sequencia eh:
       %.6f\n", n, resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio04();
    return 0;
}
```

5. Fazer:

   (a) Uma função *double parSobreImpar(int n)* que recebe um número inteiro n e retorna o n-ésimo termo da sequência $\frac{2}{3}, \frac{4}{5}, \frac{6}{7}, \frac{8}{9}, \frac{10}{11}, \ldots$. A sua função deve utilizar os 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.

```c
#include <stdio.h>

int par(int n) {
    return 2 * n;
}

double umSobreImpar(int n) {
    return 1.0 / (2 * n + 1);
}

// (a) Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
double parSobreImpar(int n) {
    return par(n) * umSobreImpar(n);
}

// (b) Função que retorna o somatório dos n primeiros termos da sequência
double somaParSobreImpar(int n) {
    double soma = 0.0;
    for (int i = 1; i <= n; i++) {
        soma += parSobreImpar(i);
    }
    return soma;
}

// (c) Procedimento exercicio05
void exercicio05() {
    int n;
    double termo, soma;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    termo = parSobreImpar(n);
    soma = somaParSobreImpar(n);

    printf("O %d-esimo termo da sequencia (2/3, 4/5, ...) eh: %.6f\n", n, termo);
    printf("O somatorio dos %d primeiros termos da sequencia eh: %.6f\n", n, soma);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio05();
    return 0;
}
```

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.

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

```c
#include <stdio.h>
#include <math.h>

// Função que retorna o n-ésimo termo da sequência de pares (2, 4, 6, 8...)
int par(int n) {
    return 2 * n;
}

// Função que retorna o n-ésimo termo da sequência 1/3, 1/5, 1/7...
double umSobreImpar(int n) {
    return 1.0 / (2 * n + 1);
}

// Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
double parSobreImpar(int n) {
    return par(n) * umSobreImpar(n);
}

// Função que retorna o n-ésimo termo da sequência com x
// (2x/3, 4x^2/5, 6x^3/7, ...)
double termoComX(int n, double x) {
    return parSobreImpar(n) * pow(x, n);
}

// Procedimento exercicio06
void exercicio06() {
    int n;
    double x, resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    printf("Digite um numero real x: ");
    scanf("%lf", &x);

    resultado = termoComX(n, x);

    printf("O %d-esimo termo da sequencia com x eh: %.6f\n", n, resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio06();
    return 0;
}
```

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.

```c
#include <stdio.h>
#include <math.h>

int par(int n) {
    return 2 * n;
}

double umSobreImpar(int n) {
    return 1.0 / (2 * n + 1);
}

// Função que retorna o n-ésimo termo da sequência 2/3, 4/5, 6/7...
double parSobreImpar(int n) {
    return par(n) * umSobreImpar(n);
}

// Função que retorna o n-ésimo termo da sequência com x
// (2x/3, 4x^2/5, 6x^3/7, ...)
double termoComX(int n, double x) {
    return parSobreImpar(n) * pow(x, n);
}

// Função que retorna o produto dos n primeiros termos da sequência
double produtoTermos(int n, double x) {
    double produto = 1.0;
    for (int i = 1; i <= n; i++) {
        produto *= termoComX(i, x);
    }
    return produto;
}

// Procedimento exercicio07
void exercicio07() {
    int n;
    double x, resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    printf("Digite um numero real x: ");
    scanf("%lf", &x);

    resultado = produtoTermos(n, x);

    printf("O produto dos %d primeiros termos da sequencia com x eh:
       %.6f\n", n, resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio07();
    return 0;
}
```

8. Fazer uma função que recebe um número inteiro n e retorna o seu fatorial[1]. Além disso, você deve fazer um procedimento exercicio08() para ler o valor de n e chamar a função desenvolvida nesta questão.

```c
#include <stdio.h>

// Função que retorna o fatorial de n
double fatorial(int n) {
    double resultado = 1.0;
    for (int i = 1; i <= n; i++) {
        resultado *= i;
    }
    return resultado;
}

// Procedimento exercicio08
void exercicio08() {
    int n;
    double resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    resultado = fatorial(n);

    printf("O fatorial de %d eh: %.0f\n", n, resultado);
}

// Função principal apenas para rodar o exercício
int main() {
    exercicio08();
    return 0;
}
```

[1]DICA: 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.

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

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

```c
#include <stdio.h>
#include <math.h>

/* Reuso das funções anteriores */
int par(int n) {
    return 2 * n;                   // 2, 4, 6, ...
}

double fatorial(int n) {            // retorna n! como double
    if (n < 0) return 0.0;          // simples proteção
    double r = 1.0;
    for (int i = 1; i <= n; i++) r *= i;
    return r;
}

/* n-ésimo termo: (2n * x^n) / (2n + 1)! */
double termoFatorialComX(int n, double x) {
    if (n < 1) return 0.0;          // sequência indexada a partir de n=1
    int numPar = par(n);            // 2n
    int fatIndice = 2 * n + 1;      // (2n+1)!
    return (numPar * pow(x, n)) / fatorial(fatIndice);
}

/* exercicio09: lê n e x e mostra o n-ésimo termo */
void exercicio09() {
    int n;
    double x;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    printf("Digite um numero real x: ");
    scanf("%lf", &x);

    double termo = termoFatorialComX(n, x);
    printf("O %d-esimo termo da sequencia eh: %.10f\n", n, termo);
}

/* main apenas para executar o exercicio */
int main() {
    exercicio09();
    return 0;
}
```

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.

```c
#include <stdio.h>
#include <math.h>

/* Reuso de funções anteriores */
int par(int n) {
    return 2 * n;   // 2, 4, 6, ...
}

double fatorial(int n) {
    if (n < 0) return 0.0;
    double r = 1.0;
    for (int i = 1; i <= n; i++) r *= i;
    return r;
}

/* n-ésimo termo: (2n * x^n) / (2n+1)! */
double termoFatorialComX(int n, double x) {
    if (n < 1) return 0.0;
    return (par(n) * pow(x, n)) / fatorial(2 * n + 1);
}

/* Somatório dos n primeiros termos */
double somaTermosFatorialComX(int n, double x) {
    double soma = 0.0;
    for (int i = 1; i <= n; i++) {
        soma += termoFatorialComX(i, x);
    }
    return soma;
}

/* Procedimento solicitado */
void exercicioXX() {
    int n;
    double x;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    printf("Digite um numero real x: ");
    scanf("%lf", &x);

    double resultado = somaTermosFatorialComX(n, x);

    printf("O somatorio dos %d primeiros termos da sequencia eh:
       %.10f\n", n, resultado);
}

/* main apenas para teste */
int main() {
    exercicioXX();
    return 0;
}
```

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.

```c
#include <stdio.h>
#include <math.h>

int par(int n) {
    return 2 * n;   // 2, 4, 6, ...
}

double fatorial(int n) {
    if (n < 0) return 0.0;
    double r = 1.0;
    for (int i = 1; i <= n; i++) r *= i;
    return r;
}

/* n-ésimo termo: (2n * x^n) / (2n+1)! */
double termoFatorialComX(int n, double x) {
    if (n < 1) return 0.0;
    return (par(n) * pow(x, n)) / fatorial(2 * n + 1);
}

/* Somatório omitindo termos cujo denominador seja múltiplo de 2 */
double somaTermosOmitindoDenominadorPar(int n, double x) {
    double soma = 0.0;
    for (int i = 1; i <= n; i++) {
        int denomIndex = 2 * i + 1;
        double denom = fatorial(denomIndex);

        if (fmod(denom, 2.0) != 0.0) { // só soma se não for múltiplo de 2
            soma += termoFatorialComX(i, x);
        }
    }
    return soma;
}

void exercicioXX() {
    int n;
    double x;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    printf("Digite um numero real x: ");
    scanf("%lf", &x);

    double resultado = somaTermosOmitindoDenominadorPar(n, x);

    printf("O somatorio (omitindo termos com denominador multiplo de 2) eh: %.10f\n", resultado);
}

int main() {
    exercicioXX();
    return 0;
}
```

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.

```c
#include <stdio.h>

/* Função que imprime os n primeiros múltiplos de 5 */
void multiplosDe5(int n) {
    printf("Os %d primeiros multiplos de 5 sao:\n", n);
    for (int i = 1; i <= n; i++) {
        printf("%d ", 5 * i);
    }
    printf("\n");
}

/* Procedimento solicitado */
void exercicioXX() {
    int n;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    multiplosDe5(n);
}

/* Função principal para rodar o exercício */
int main() {
    exercicioXX();
    return 0;
}
```

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.

```c
#include <stdio.h>

/* Função que retorna o n-ésimo termo da sequência de Fibonacci */
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

/* Procedimento solicitado */
void exercicioXX() {
    int n, resultado;

    printf("Digite um numero inteiro n: ");
    scanf("%d", &n);

    resultado = fibonacci(n);

    printf("O %d-esimo termo da sequencia de Fibonacci eh: %d\n", n,
       resultado);
}

/* Função principal apenas para rodar o exercício */
int main() {
    exercicioXX();
    return 0;
}
```

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.

```c
#include <stdio.h>

// Função que retorna o maior elemento de Fibonacci menor que n
int maiorFibonacciMenorQueN(int n) {
    int a = 0, b = 1, prox = 0;

    // Caso n seja menor ou igual a 1, não existe Fibonacci menor que n
    if (n <= 1) {
        return -1;
    }

    while (b < n) {
        prox = a + b;
        a = b;
        b = prox;
    }

    return a; // a será o maior Fibonacci menor que n
}

// Procedimento solicitado
void exercicioXX() {
    int n, resultado;

    printf("Digite um numero inteiro: ");
    scanf("%d", &n);

    resultado = maiorFibonacciMenorQueN(n);

    if (resultado == -1) {
        printf("Nao existe numero de Fibonacci menor que %d.\n", n);
    } else {
        printf("O maior numero de Fibonacci menor que %d eh: %d\n", n,
            resultado);
    }
}

int main() {
    exercicioXX();
    return 0;
}
```