Professor Pietro Martins de Oliveira

ALGORITMOS

do início ao fim

PACOTE DE EXERCÍCIOS 4: ESTRUTURAS DE REPETIÇÃO EM C

1) Desenvolva um algoritmo que some todos os números inteiros compreendidos entre 1 e 10 (inclusive).

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 1 - Caso de teste
Somatório:  55
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 1 - Solução

```c
#include <stdio.h>
int main(){
    int soma, i;
    soma = 0;
    i = 1;
    while(i<=10){
        soma = soma + i; //soma += i
        i = i + 1; // i++
    }
    printf("Somatório: %d.\n", soma);
}
```

2) Desenvolva um algoritmo que receba dois números inteiros positivos A e B. Exiba na tela todos os números inteiros compreendidos entre A e B, excluindo os próprios A e B. Suponha que o usuário respeite o enunciado e insira valores válidos para A e B.

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 2 - Caso de teste
Insira o valor de A:
1
Insira o valor de B:
10
Série numérica:
 2  3  4  5  6  7  8  9
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 2 - Solução

```c
#include <stdio.h>
int main(){
    int A, B, i;
    printf("Insira o valor de A:\n");
    scanf("%d", &A);
    printf("Insira o valor de B:\n");
    scanf("%d", &B);
    i = A+1;
    printf("Série numérica:\n");
    while(i < B){
        printf("%d ", i);
        i++;
    }
}
```

3) Desenvolva um algoritmo que receba um número N e calcule o fatorial de N, sabendo que N! = N * (N-1) * (N-2) * ... * 3 * 2 * 1.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

Exemplo de execução - Exercício 3 - Caso de teste

```
Insira o valor de N:
5
O fatorial de  5 é  120
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 3 - Solução

```c
#include <stdio.h>
int main(){
    int i, N, fat;
    printf("Insira o valor de N:\n");
    scanf("%d", &N);
    fat = 1;
    for(i=N; i>=1; i--){
        fat = fat * i;
    }
```

```
10      printf("O fatorial de %d é %d.\n", N, fat);
11  }
```

4) Desenvolva um algoritmo que receba um número N e imprima a tabuada de N, na tela.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 4 - Caso de teste
Insira N:
5
 5 x  1 =  5
 5 x  2 =  10
 5 x  3 =  15
 5 x  4 =  20
 5 x  5 =  25
 5 x  6 =  30
 5 x  7 =  35
 5 x  8 =  40
 5 x  9 =  45
 5 x  10 =  50
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
ESTRUTURAS DE REPETIÇÃO - Exercício 4 - Solução
01  #include <stdio.h>
02  int main(){
03      int N, i, res;
04      printf("Insira N:\n");
05      scanf("%d", &N);
06      i = 1;
07      while(i <= 10){
08          res = N*i;
09          printf("%d x %d = %d\n", N, i, res);
10          i++;
11      }
12  }
```

5) Charlinho tem 11 anos, mede 1,40 metros de altura e cresce em média 2,1 centímetros ao ano. Seu irmão, Bossa, aos 14 anos, tem 1,45 metros de altura e cresce em média 1,1 centímetro por ano. Elabore um programa que conte quantos anos serão necessários para que a altura de Charlinho ultrapasse a de Bossa.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 5 - Caso de teste
Serão necessários:  6 anos
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO – Exercício 5 – Solução

```c
#include <stdio.h>
int main(){
    float C, B;
    int anos;
    anos = 0;
    C = 140;
    B = 145;
    while(C <= B){
        anos++;
        C += 2.1;
        B += 1.1;
    }
    printf("Serão necessários %d anos.\n", anos);
}
```

6) (Adaptado de ASCENCIO e CAMPOS, 2008) Um funcionário de uma empresa recebe aumento salarial anualmente. Sabe-se que:

a) esse funcionário foi contratado em 2015, com salário inicial de R$ 1.000,00;
b) em 2016 recebeu aumento de 1,5% sobre seu salário inicial;
c) a partir de 2017 (inclusive), os aumentos salariais sempre corresponderam ao dobro da porcentagem do ano anterior.

Faça um programa que receba o ano atual determine o salário atual desse funcionário.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em

branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 6 - Caso de teste
Qual é o ano atual?
2020
O salário atual é:  1539.0362176
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
ESTRUTURAS DE REPETIÇÃO - Exercício 6 - Solução
01 #include <stdio.h>
02 int main(){
03     float sal, sal_novo, porc;
04     int i, ano_atual;
05 
06     printf("Qual é o ano atual?\n");
07     scanf("%d", &ano_atual);
08     sal = 1000.0;
09     porc = 1.5/100.0;
10     sal_novo = sal + sal*porc;
11     for(i=2017;i<=ano_atual;i++){
12         porc = 2 * porc;
13         sal_novo = sal_novo + sal_novo*porc;
14     }
15     printf("O salário atual é: %.2f.\n", sal_novo);
16 }
```

7) Desenvolva um algoritmo que peça para o usuário inserir vários números inteiros. O algoritmo deverá contabilizar a quantidade de números positivos informados. Caso o usuário digite 0, o algoritmo deve mostrar quantidade contabilizada e encerrar.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 7 - Caso de teste
Insira um número:
5
Insira um número:
5
Insira um número:
-1
Insira um número:
```

```
0
Quantidade de positivos:  2
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO – Exercício 7 – Solução

```c
#include <stdio.h>
int main(){
    int num, cont;
    cont = 0;
    do{
        printf("Insira um número:\n");
        scanf("%d", &num);
        if(num > 0){
            cont++;
        }
    }while(num != 0);
    printf("Quantidade de positivos: %d\n", cont);
}
```

8) Desenvolva um algoritmo que peça para o usuário informar dois números. Após isso, o algoritmo deve mostrar cálculo o primeiro número elevado ao segundo. Ao final, o algoritmo deve perguntar se o usuário deseja repetir a operação. Caso o usuário insira o caractere "s", o algoritmo solicita novamente dois números e mostra novamente a potência do primeiro pelo segundo. Caso contrário, o algoritmo é encerrado.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

Exemplo de execução – Exercício 8 - Caso de teste

```
Insira o primeiro número:
2
Insira o segundo número:
3
A elevando a B:  8

Deseja repetir a operação? (s/n)
s

Insira o primeiro número:
2
Insira o segundo número:
2
```

```
A elevando a B:  4

Deseja repetir a operação? (s/n)
n
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO – Exercício 8 – Solução

```c
#include <stdio.h>
#include <math.h>
int main(){
    float A, B, res;
    char op;
    do{
        printf("Insira o primeiro número:\n");
        scanf("%f", &A);
        fflush(stdin);
        printf("Insira o segundo número:\n");
        scanf("%f", &B);
        fflush(stdin);
        res = pow(A, B);
        printf("A elevado a B: %f.\n", res);
        printf("Deseja repetir a operação? (tecle S para sim)\n");
        scanf("%c", &op);
        fflush(stdin);
    }while(op == 'S' || op == 's');
}
```

9) Desenvolva um algoritmo que peça ao usuário que insira dois números inteiros positivos A e B, no qual A deve ser menor que B (supõe-se que o usuário irá respeitar esse enunciado). O algoritmo deve mostrar, na tela, todos os números ímpares compreendidos entre A e B (inclusive).

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

Exemplo de execução – Exercício 9 - Caso de teste

```
Insira o valor de A:
1
Insira o valor de B:
10
É impar: 1
```

```
É impar: 3
É impar: 5
É impar: 7
É impar: 9
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 9 - Solução

```c
#include <stdio.h>
int main(){
    int A, B, i;
    printf("Insira o valor de A:\n");
    scanf("%d", &A);
    printf("Insira o valor de B:\n");
    scanf("%d", &B);
    for(i=A; i<=B; i++){
        if((i % 2) != 0){
            printf("É ímpar: %d.\n", i);
        }
    }
}
```

10) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que leia dez conjuntos de dois valores, o primeiro representando o número do aluno e o segundo representando sua altura em centímetros. Encontre o aluno mais alto e o mais baixo. Mostre o número do aluno mais alto e o número do mais baixo, junto com suas alturas.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

Exemplo de execução - Exercício 10 - Caso de teste

```
Insira o número do aluno:
1
Insira a altura do aluno:
1,90
Insira o número do aluno:
2
Insira a altura do aluno:
1,91
Insira o número do aluno:
3
Insira a altura do aluno:
1,92
```

```
Insira o número do aluno:
4
Insira a altura do aluno:
1,99
Insira o número do aluno:
5
Insira a altura do aluno:
1,98
Insira o número do aluno:
6
Insira a altura do aluno:
1,64
Insira o número do aluno:
7
Insira a altura do aluno:
1,69
Insira o número do aluno:
8
Insira a altura do aluno:
1,56
Insira o número do aluno:
9
Insira a altura do aluno:
1,66
Insira o número do aluno:
10
Insira a altura do aluno:
1,89
Número do maior aluno:  4
Altura do maior aluno:  1.99
Número do menor aluno:  8
Altura do menor aluno:  1.56
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 10 - Solução

```c
#include <stdio.h>
int main(){

    int i, num, n_maior, n_menor;
    float alt, maior, menor;

    for(i=1; i<=10; i++){
        printf("Insira o número do aluno:\n");
        scanf("%d", &num);
        fflush(stdin);
        printf("Insira a altura do aluno:\n");
        scanf("%f", &alt);
        fflush(stdin);
        if(i == 1){
```

```
15              maior = alt;
16              n_maior = num;
17              menor = alt;
18              n_menor = num;
19          } else {
20              if(alt > maior){
21                  maior = alt;
22                  n_maior = num;
23              }
25              if(alt < menor){
26                  menor = alt;
27                  n_menor = num;
28              }
29          }
30      }
31      printf("Número do maior aluno: %d.\n", n_maior);
32      printf("Altura do maior aluno: %.2f.\n", maior);
33      printf("Número do menor aluno: %d.\n", n_menor);
34      printf("Altura do menor aluno: %.2f.\n", menor);
35  }
```

11) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que mostre os oito primeiros termos da sequência de Fibonacci.

0-1-1-2-3-5-8-13-21-34-55-...

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 11 - Caso de teste
Série de Fibonacci:
 0  1  1  2  3  5  8  13
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
ESTRUTURAS DE REPETIÇÃO - Exercício 11 - Solução
01  #include <stdio.h>
02  int main(){
03      int i, ant1, ant2, atual;
04      ant1 = 1;
05      ant2 = 0;
06      printf("Série de Fibonacci (8 primeiros termos):\n");
07      printf("%d ", ant2);
08      printf("%d ", ant1);
```

```
09      for(i=3; i<=8; i++){
10          atual = ant1 + ant2;
11          printf("%d ", atual);
12          ant2 = ant1;
13          ant1 = atual;
14      }
15  }
```

12) (Adaptado de ASCENCIO e CAMPOS, 2008) Desenvolva um algoritmo que receba um número N e informe se N é um número primo, ou não. A saber: um número primo é um inteiro positivo que só pode ser dividido por ele mesmo e por um, apenas.

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, nos quais você encontra simulações da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 12 - Caso de teste XXX
Insira um número:
7
 7 é primo.
```

```
Exemplo de execução – Exercício 12 - Caso de teste XXX
Insira um número:
10
 10 não é primo.
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
ESTRUTURAS DE REPETIÇÃO – Exercício 12 – Solução
01  #include <stdio.h>
02  int main(){
03      int N, i, cont;
04      printf("Insira um número:\n");
05      scanf("%d", &N);
06      cont = 0;
07      for(i=1; i<=N; i++){
08          if((N % i) == 0){
09              cont++;
10          }
11      }
12      if(cont == 2){
13          printf("%d é primo.\n", N);
14      }
15      else {
```

```
16          printf("%d não é primo.\n", N);
17      }
18  }
```

13) Desenvolva um algoritmo que mostre a tabuada de todos os números inteiros compreendidos entre 1 e 10 (inclusive).

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 13 - Caso de teste
 1 x  1 =  1
 1 x  2 =  2
 1 x  3 =  3
 1 x  4 =  4
 1 x  5 =  5
 1 x  6 =  6
 1 x  7 =  7
 1 x  8 =  8
 1 x  9 =  9
 1 x  10 =  10

 2 x  1 =  2
 2 x  2 =  4
 2 x  3 =  6
 2 x  4 =  8
 2 x  5 =  10
 2 x  6 =  12
 2 x  7 =  14
 2 x  8 =  16
 2 x  9 =  18
 2 x  10 =  20

 3 x  1 =  3
 3 x  2 =  6
 3 x  3 =  9
 3 x  4 =  12
 3 x  5 =  15
 3 x  6 =  18
 3 x  7 =  21
 3 x  8 =  24
 3 x  9 =  27
 3 x  10 =  30

 4 x  1 =  4
```

```
4 x  2 =  8
4 x  3 =  12
4 x  4 =  16
4 x  5 =  20
4 x  6 =  24
4 x  7 =  28
4 x  8 =  32
4 x  9 =  36
4 x  10 =  40

5 x  1 =  5
5 x  2 =  10
5 x  3 =  15
5 x  4 =  20
5 x  5 =  25
5 x  6 =  30
5 x  7 =  35
5 x  8 =  40
5 x  9 =  45
5 x  10 =  50

6 x  1 =  6
6 x  2 =  12
6 x  3 =  18
6 x  4 =  24
6 x  5 =  30
6 x  6 =  36
6 x  7 =  42
6 x  8 =  48
6 x  9 =  54
6 x  10 =  60

7 x  1 =  7
7 x  2 =  14
7 x  3 =  21
7 x  4 =  28
7 x  5 =  35
7 x  6 =  42
7 x  7 =  49
7 x  8 =  56
7 x  9 =  63
7 x  10 =  70

8 x  1 =  8
8 x  2 =  16
8 x  3 =  24
8 x  4 =  32
8 x  5 =  40
8 x  6 =  48
8 x  7 =  56
8 x  8 =  64
8 x  9 =  72
8 x  10 =  80
```

```
9 x  1 =  9
9 x  2 =  18
9 x  3 =  27
9 x  4 =  36
9 x  5 =  45
9 x  6 =  54
9 x  7 =  63
9 x  8 =  72
9 x  9 =  81
9 x  10 =  90

10 x  1 =  10
10 x  2 =  20
10 x  3 =  30
10 x  4 =  40
10 x  5 =  50
10 x  6 =  60
10 x  7 =  70
10 x  8 =  80
10 x  9 =  90
10 x  10 =  100
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 13 - Solução

```c
#include <stdio.h>
int main(){
    int i, j, res;
    i=1;
    while(i <= 10){
        j=1;
        while(j <= 10){
            res = i*j;
            printf("%d x %d = %d\n", i, j, res);
            j++;
        }
        printf("\n\n");
        i++;
    }
}
```

14) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que leia um número N e que indique quantos valores inteiros e positivos devem ser lidos a seguir. Para cada número lido, mostre o fatorial desse valor.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens

geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 14 - Caso de teste
Quantos números serão informados:
3

Insira o numero  1
5
O fatorial de  5 é  120

Insira o numero  2
4
O fatorial de  4 é  24

Insira o numero  3
3
O fatorial de  3 é  6
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO – Exercício 14 – Solução

```c
#include <stdio.h>
int main(){
    int N, fat, i, j, num;
    printf("Quantos números serão informados:\n");
    scanf("%d", &N);
    for(i=1; i<=N; i++){
        printf("Insira o número %d:\n", i);
        scanf("%d", &num);
        fat = 1;
        for(j=1; j<=num; j++){
            fat = fat * j;
        }
        printf("O fatorial de %d é %d.\n", num, fat);
    }
}
```

15) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que leia um valor N inteiro e positivo, calcule e mostre o valor de E, conforme a fórmula a seguir:

E = 1 + 1/1! + 1/2! + 1/3! + ... + 1/N!

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens

geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 15 - Caso de teste
Insira o valor de N:
5
Resultado:  2.71666666666667
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO – Exercício 15 – Solução

```c
#include <stdio.h>
int main(){
    int N, i, j;
    float res, fat;
    printf("Insira o valor de N:\n");
    scanf("%d", &N);
    res = 1;
    for(i=1; i<=N; i++){
        fat = 1;
        for(j=1; j<=i; j++){
            fat = fat * (float)j;
        }
        res = res + 1/fat;
    }
    printf("Resultado: %f.\n", res);
}
```

16) (Adaptado de ASCENCIO e CAMPOS, 2008) Desenvolva um algoritmo que peça ao usuário que insira cinco conjuntos de dois números inteiros positivos (A, B), no qual A deve ser menor que B (supõe-se que o usuário irá respeitar esse enunciado). Para cada dupla (A, B), informada pelo usuário, o algoritmo deve mostrar, na tela, todos os números pares compreendidos entre A e B (inclusive).

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 16 - Caso de teste
Insira um valor de A:
1
Insira um valor de B:
5
É par: 2
```

```
É par: 4

Insira um valor de A:
6
Insira um valor de B:
10
É par: 6
É par: 8
É par: 10

Insira um valor de A:
11
Insira um valor de B:
15
É par: 12
É par: 14

Insira um valor de A:
16
Insira um valor de B:
19
É par: 16
É par: 18

Insira um valor de A:
20
Insira um valor de B:
21
É par: 20
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

ESTRUTURAS DE REPETIÇÃO - Exercício 16 - Solução

```c
#include <stdio.h>
int main(){
    int A, B, i, j;
    for(i=1; i<=5; i++){
        printf("Insira um valor para A:\n");
        scanf("%d", &A);
        printf("Insira um valor para B:\n");
        scanf("%d", &B);
        for(j=A; j<=B; j++){
            if((j % 2) == 0){
                printf("%d é par.\n", j);
            }
        }
    }
}
```