**Exercícios de Algoritmos**

**1.**

**Pseudo-Código**

**Algoritmo LTO1\_EstSeq01**

Declarar

Lado, área : int;

Inicio

Ler lado

Area = (lado \* lado);

Mostrar área

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Lado** | **5** |
| **Área** | **25** |

**Mesa de dados Saída**

5 25

**2.**

**Pseudo-Código**

**Algoritmo SalNovo**

Declarar

Salario, reajuste:Real;

Inicio

Ler salario

Reajuste = (salario x 1,15);

Mostrar reajuste

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Salário** | **1000** |
| **Reajuste** | **1150** |

**Mesa de dados Saída**

1000 1150

**3.**

**Pseudo-Código**

**Algoritmo Exerc3**

Declarar

Base, altura, área: Real

Inicio

Ler base;

Ler altura;

Área = ((base \* altura)/2);

Mostrar área

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Base** | **2** |
| **Altura** | **4** |
| **Área** | **4** |

**Mesa de dados Saída**

2 4

4

**4.**

**Pseudo-Código**

**Algoritmo Celsius**

Declarar

C,F;real

Inicio

Ler C

F=((9\*C+160)/5)

Mostrar F

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **C** | **28** |
| **F** | **82,4** |

**Mesa de dados Saída**

28 82,4

**5.**

**Pseudo-Código**

**Algoritmo Equação**

Declarar

A,B,C,R1,R2: real

Inicio

Ler(A);

Ler(B);

Ler(C);

r1=(-B+raiz((B\*B)-4\*A\*C))/2\*A;

r2=(-B-raiz((B\*B)-4\*A\*C))/2\*A;

Mostrar r1;

Mostrar r2

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **A** | **2** |
| **B** | **-9** |
| **C** | **5** |
| **R1** | **15** |
| **R2** | **3** |

**Mesa de dados Saída**

**A =** 2 15

**B =** 3 3

**C =** 4

**6.**

**Pseudo-Código**

**Algoritmo XY**

Declarar

X,Y,X1,Y1 : real;

Início.

Ler(X);

Ler(Y);

X1=Y;

Y1=X;

Mostra X,Y

Fim

**TESTE DE MESA**

|  |  |  |
| --- | --- | --- |
| **X** | **1** | **2** |
| **Y** | **2** | **1** |

**Mesa de dados Saída**

1 2

2 1

**7.**

**Pseudo-Código**

**Algoritmo volume**

Declarar.

C, L, A, V: real;

Início.

Ler C;

Ler L;

Ler A;

V=(C\*L\*A);

Mostrar V

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **C** | **2** |
| **L** | **3** |
| **A** | **4** |
| **V** | **24** |

**Mesa de dados Saída**

2 24

3

4

**8.**

**Pseudo-Código**

**Algoritmo PoupRend**

Declarar

P: num;

Início.

Ler P;

PoupRend = (P +((P/100)\*1,3));

Mostrar PoupRend

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **P** | **500** |
| **PoupRend** | **506,50** |

**Mesa de dados Saída**

500 506,50

**9.**

**Pseudo-Código**

**Algoritmo Quadrado**

Declarar

X, Y, Soma: int;

Início.

Ler X;

Ler Y;

X = (X \* X);

Y = (Y \* Y);

Soma = (X + Y);

Mostrar Soma

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **X** | **3** |
| **Y** | **3** |
| **Soma** | **18** |

**Mesa de dados Saída**

3 18

3

**10.**

**Pseudo-Código**

**Algoritmo Diferença**

Declarar

X, Y, D: reais;

Início

Ler X;

Ler Y;

Se (X > Y) então

R = (X-Y);

Se (X==Y) então

R=(0);

Se(Y<X) então

R=(Y-X);

Mostrar R

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **X** | **7** |
| **Y** | **7** |
| **D** | **0** |

**Mesa de dados Saída**

7 0

7

**11.**

**Pseudo-Código**

Algoritmo Pi

Declarar

R: real;

Inicio

Ler R;

C = (2\*3,14\*R);

Mostrar C

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **R** | **5** |
| **C** | **31,4** |

**Mesa de dados Saída**

5 31,4

**12.**

**Pseudo-Código**

**Algoritmo idadFut**

Declarar

Ano1, ano2, idade1, idadeFutura: int;

Início

Ler ano1;

Ler ano2;

Idade1 = (ano2-ano1);

idadeFutura = (idade1+17);

Mostrar idadeFutura

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Ano1** | **1991** |
| **Ano2** | **2019** |
| **Idade1** | **28** |
| **IdadeFutura** | **45** |

**Mesa de dados Saída**

199145

2019

**13.**

**Pseudo-Código**

**Algoritmo Dieta**

Declarar

kg: real;

Início

Ler kg;

Dias = ((kg\*1000)/50)

Mostrar dias

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Kg** | **2** |
| **Dias** | **40** |

**Mesa de dados Saída**

2 40

**14.**

**Pseudo-Código**

**Algoritmo angulo**

Declarar

ang1, ang2, ang3: real;

Início

Ler ang1;

Ler ang2;

ang3 = (180 – (ang1 + ang2));

Mostrar ang3

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Ang1** | **80** |
| **Ang2** | **65** |
| **Ang3** | **35** |

**Mesa de dados Saída**

80 35

65

**15.**

**Pseudo-Código**

**Algoritmo hipotenusa**

Declarar

cat1, cat2,h: real;

Início

Ler cat1;

Ler cat2;

h=(sqr((cat1\*cat1)+(cat2\*cat2)));

Mostrar h

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Cat1** | **9** |
| **Cat2** | **12** |
| **Hipo** | **15** |

**Mesa de dados Saída**

9 15

12

**16.**

**Pseudo-Código**

**Algoritmo sal\_liq**

Declarar

H, V, DESC, DEP, SALB, SAL: real;

Início

Ler H;

Ler V;

Ler DESC;

Ler DEP;

Salb = ht\*vh;

Sal = (salb – (salb\*desc/100) + (dep\*100));

Mostra Sal

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **H** | **44** |
| **V** | **11** |
| **Desc** | **2** |
| **Dep** | **2** |
| **SalBru** | **484** |
| **Sal** | **674,32** |

**Mesa de dados Saída**

44 674,32

11

2

2

**17.**

**Pseudo-Código**

**Algoritmo gasolina**

Declarar

Litros, vMedia, distancia, tempo: num;

Início.

Ler tempo;

Ler velocidade;

Distancia = (tempo\*vMedia);

Litros = (distancia/12);

Mostrar litros

Fim

**TESTE DE MESA**

|  |  |
| --- | --- |
| **Tempo** | **3** |
| **vMedia** | **60** |
| **Distancia** | **180** |
| **Litro** | **15** |

**Mesa de dados Saída**

3 15

60

**18.** **Receba 2 valores inteiros. Calcule e mostre o resultado da diferença do maior**

**pelo menos valor.**

**Algoritmo da diferença**

Declarar

X,Y: int;

Inicio

Ler(X);

Ler(Y);

Se(X>=Y) então

X=(X-Y);

Mostrar X;

Senão

X=(Y-X);

Mostrar X;

Fim-se

Fim

T**este de Mesa**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | 10 | 5 | 10 |
| **Y** | 10 | 10 | 5 |

**Mesa de dados Saída**

10 5 10 0

10 10 5 5

5

**19. Receba 2 valores reais. Calcule e mostre o maior deles.**

**Algoritmo Maior**

Declarar

X, Y: Int

Inicio

Ler X;

Ler Y;

Se (X>=Y) então

X = (X-Y);

Mostrar X;

Senão

Y=(Y-X);

Mostrar Y;

Fim – se

Fim

**Mesa de dados**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | 1 | 2 | 3 |
| **Y** | 3 | 2 | 1 |

**Massa de dados Saída**

1 2 3 2

3 2 1 0

2

**20.** **Receba 3 coeficientes A, B, e C de uma equação do 2º grau da fórmula AX²+BX+C=0. Verifique e mostre a existência de raízes reais e se caso exista, calcule e mostre.**

**Algoritmo grau**

Declarar

a, b, c, d, x1, x2: real;

Inicio

Ler(a);

Ler(b);

Ler(c);

Se (a==0) então

Mostrar (“não é uma equação de segundo grau”);

Senão

D = ((B\*B) - 4 \* A \*C);

Se (D<0) então

Mostra (“não há raíz real”);

Senão

Se (D==0)

x1= ((-b)/2.a);

x2=x1;

Mostrar (“há apenas uma raiz real”)

Senão

x1=((-b+sqr d)/2.a);

x2=((-b-sqr d)/2.a);

fim-se;

fim-se;

Fim-se;

Fim;

**Teste de Mesa**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | 1 | 1 | 4 | 1 |
| **B** | 3 | 1 | 4 | 5 |
| **C** | 4 | 1 | 1 | 6 |
| **D** |  | -3 | 0 | 1 |
| **X1** |  |  | -8 | -2 |
| **X2** |  |  | -8 | -3 |

**Massa de dados Saída**

0 3 4 “não é uma equação de segundo grau”

1 1 1 “não há raíz real”

4 4 1 “há apenas uma raiz real” -8 -8

1 5 6 -2 -3

**21.** **Receba 4 notas bimestrais de um aluno. Calcule e mostre a média aritmética. Mostre a mensagem de acordo com a média:**

**a. Se a média for >= 6,0 exibir “APROVADO”;**

**b. Se a média for >= 3,0 ou < 6,0 exibir “EXAME”;**

**c. Se a média for < 3,0 exibir “RETIDO”.**

**Algoritmo Notas**

Declarar

N1, n2, n3, n4: real;

Inicio

Ler N1;

Ler N2;

Ler N3;

Ler N4;

Media = ((n1+n2+n3+n4))/4);

Se (media<3) então

Mostrar (“Retido”);

Senão

Se (media >=6) então

Mostrar (“Aprovado”);

Senão

Mostrar (“Exame”);

Fim-se;

Fim-Se;

Fim

**Teste de mesa**

|  |  |  |  |
| --- | --- | --- | --- |
| **N1** | 2 | 6 | 1 |
| **N2** | 4 | 6 | 1 |
| **N3** | 6 | 6 | 1 |
| **N4** | 9 | 6 | 1 |
| **Media** | 5,25 | 6 | 1 |

**Massa de dados Saída**

2 6 1 “Exame”

4 6 1 “Aprovado”

6 6 1 “Retido”

9 6 1

**22. Receba 2 valores inteiros e diferentes. Mostre seus valores em ordem crescente.**

**Algoritmo crescente**

Declarar

X, Y: int;

Inicio

Ler X

Ler Y

Se (x<=y) então

Mostrar X;

Mostrar Y;

Senão

Mostrar Y;

Mostrar X;

Fim-se

Fim

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | 1 | 2 | 0 |
| **Y** | 2 | 1 | 0 |

**Massa de dados Saída**

1. 2 0 1 2

2 1 0 1 2

0 0

**23. Receba 3 valores obrigatoriamente em ordem crescente e um 4º valor não necessariamente em ordem. Mostre os 4 números em ordem crescente.**

**Algoritmo Crescente2**

Declarar

C1, C2, C3, C4: real;

Inicio

Ler C1;

Ler C2;

Ler C3;

Ler C4;

Se (C1>C4)

Mostrar C4;

Mostrar C1;

Mostrar C2;

Mostrar C3;

Senão

Mostrar C1;

Se (C2>C4) então

Mostrar C4;

Mostrar C2;

Mostrar C3;

Senão

Mostrar C1;

Mostrar C2;

Se (C3>C4) então

Mostrar C4;

Mostrar C3;

Senão

Mostrar C3;

Mostrar C4

Fim -se

Fim-se;

Fim-se

Fim

**Teste de mesa**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C1** | 1 | 5 | 1 |  |
| **C2** | 2 | 6 | 3 |  |
| **C3** | 3 | 7 | 5 |  |
| **C4** | 4 | 1 | 4 |  |

**Massa de dados Saída**

1 5 1 1 2 3 4

2 6 3 1 5 6 7

3 7 5 1 3 4 5

4 1 4

**24. Receba um valor inteiro. Verifique e mostre se é divisível por 2 e 3.**

**Algoritmo Inteiro**

Declarar

X, R1, R2: Int;

Inicio

Ler X;

R1= X%2;

R2=X%3;

Se (R1==0 &&R2==0)

Mostrar X (“é divisível por 2”)

Senão

Se (R2==0)

Mostrar X (“é divisível por 3”);

Senão

Mostrar X (“Não é divisível por 2 e nem por 3”);

Fim-Se

Fim-Se

Fim-se

Fim

**Teste de Mesa**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | 4 | 9 | 47 |
| **R1** | 0 | 1 | 1 |
| **R2** | 1 | 0 | 2 |

**Massa de dados Saída**

4 9 47 "Divisível apenas por 2"

"Divisível apenas por 3"

"Não divisível por 2 nem 3"

**25. Receba a hora de início e de final de um jogo (HH,MM) sabendo que o tempo máximo é de 24 horas e pode começar num dia e terminar noutro.**

**Algoritmo Jogo**

Declarar

Hr1, hrF, min1, minF, tempoH, tempoM :int;

Inicio

Escrever (“Digite inicio de jogo”)

Ler (hr1);

Ler (min1);

Escrever (“Digite tempo final de jogo”)

Ler (hrF);

Ler (minF);

Se (hrF>=hr1)

tempoh= (hrF-hr1);

senão

tempoh= (hrF+24-h1);

fim-se;

Se(mF>=mi)

tempom= (minF-min1);

senão

tempom= (minF+59-min1);

fim-se;

Se (hi<=0 || hF<=0 || hi>24 || hF>24 || mi<=0 ||mF<=0 || mi>59 || mF>59)

Mostra (“ERRO valor inválido”);

Senão

Mostra (“o Jogo durou” tempom “horas e” tempom “minutos”);

Fim-se;

Fim

Teste de Mesa

Hi

0

22

4

20

30

Mi

0

0

4

50

10

hF

2

2

8

2

10

mF

50

50

0

10

10

Tempoh

2

4

4

6

20

Tempom

50

50

55

19

|  |  |  |  |
| --- | --- | --- | --- |
| Hr1 | 23 | 23 | 03 |
| Min1 | 59 | 00 | 30 |
| HrF | 02 | 02 | 06 |
| MinF | 00 | 59 | 00 |
| TempoH | 2 | 3 | 2 |
| TempoM | 01 | 59 | 30 |

**Massa de Dados Saída**

23 59 02 00 “o Jogo durou 2 horas e 01 minutos”

23 00 02 59 “o Jogo durou 3 horas e 59 minutos”

03 30 06 00 “o Jogo durou 2 horas e 30 minutos”

**26.**

**Algorítmo ex\_26**

Declarar

x, y, r:Int;

Inicio

Ler x;

Ler y;

Se (x>=y) então

r = x % y;

Senão

r = y % x;

Fim-Se

Se (r == 0) então

Mostra "O maior é múltiplo do menor";

Senão

Mostra "O maior não é múltiplo do menor";

Fim-Se

Fim

**Teste de mesa**

|  |  |  |
| --- | --- | --- |
| **X** | 3 | 4 |
| **Y** | 2 | 8 |
| **R** | 1 | 0 |

**Massa de dados Saída**

3 2 "O maior não é múltiplo do menor"

4 8 "O maior é múltiplo do menor"

**Mod(Procedimento)**

**Algorítmo ex\_26**

Declarar

x, y, r:Int;

Inicio

Ler x;

Ler y;

Proc\_Multiplo();

Fim

**Procedimento Proc\_Multiplo()**

Se (x>=y) então

r = x % y;

Senão

r = y % x;

Fim-Se

Se (r == 0) então

Mostra "O maior é múltiplo do menor";

Senão

Mostra "O maior não é múltiplo do menor";

Fim-Se

Segue

**27.**

Algorítmo circuito

Declarar

nV, eC, t, vM:int;

Inicio

Ler nV;

Ler eC;

Ler t;

vM = (nV \* eC) / t;

vM = (vM \* 60) / 1000;

Mostra vM;

Fim

**Teste de Mesa**

|  |  |  |
| --- | --- | --- |
| **nV** | 5 |  |
| **eC** | 500 |  |
| **T** | 10 |  |
| **vM** |  |  |

Debug

nV | 5 |

eC | 500 |

t | 10 |

vM | 250 | 15

Massa de Dados

5, 500, 10

Saídas

15

27 - Mod(Procedimento)

Algorítmo Lot01\_Dec27

Declarar

nV, eC, t, vM:Num;

Inicio

Ler nV;

Ler eC;

Ler t;

Proc\_VM();

Mostra vM;

Fim

Procedimento Proc\_VM();

vM = (nV \* eC) / t;

vM = (vM \* 60) / 1000;

Segue

**28.**

Algorítmo Lot01\_Dec28

Declarar

precoA, mediaM, precoN:Num;

Inicio

Ler mediaM;

Ler precoA;

Se (mediaM < 500 && precoA < 30) então

precoN = precoA \* 1.10;

Senao

Se(mediaM >= 500 && mediaM < 1000 && precoA >=30 && precoA <80) então

precoN = precoA \* 1.15;

Senao

Se(mediaM >= 1000 && precoA >= 80) então

precoN = precoA \* 0.95;

Senao

precoN = precoA;

Fim-Se

Fim-Se

Fim-Se

Mostra precoN;

Fim

Debug

mediaM | 90 | 500 | 999 | 1000 | 8574 | 550

precoA | 10 | 30 | 40 | 80 | 100 | 10

precoN | 11 | 34.5 | 46 | 76 | 95 | 10

Massa de Dados

90, 10

500, 30

999, 40

1000, 80

8574, 100

550, 10

Saída

11

(34.5)

46

76

95

10

28 - Mod(Procedimento)

Algorítmo Lot01\_Dec28

Declarar

precoA, mediaM:Num;

Inicio

Ler mediaM;

Ler precoA;

Proc\_NVPreco();

Fim

Procedimento Proc\_NVPreco();

Se (mediaM < 500 && precoA < 30) então

precoN = precoA \* 1.10;

Senao

Se(mediaM >= 500 && mediaM < 1000 && precoA >=30 && precoA <80) então

precoN = precoA \* 1.15;

Senao

Se(mediaM >= 1000 && precoA >= 80) então

precoN = precoA \* 0.95;

Senao

precoN = precoA;

Fim-Se

Fim-Se

Fim-Se

Mostra precoN;

Segue

**29.**

Algorítmo Lot01\_Dec29

Declarar

tipo, valor, resultado:Num;

Inicio

Ler tipo;

Ler valor;

Se (tipo == 1) então

resultado = valor \* 1,03;

Senão

Se (tipo == 2) então

resultado = valor \* 1,05;

Senão

Mostra "Tipo desconhecido, insira 1 para poupança ou 2 para renda fixa";

Fim-se

Fim-se

Mostra resultado;

Fim

Debug

tipo | 0 | 1 | 2

valor | 100 | 100 | 100

resultado | | 103 | 105

Massa de Dados

0, 100

1, 100

2, 100

Saída

"Tipo desconhecido, insira 1 para poupança ou 2 para renda fixa";

103

105

29 - Mod(Procedimento)

Algorítmo Lot01\_Dec29

Declarar

tipo, valor, resultado:Num;

Inicio

Ler tipo;

Ler valor;

Proc\_CCPP();

Fim

Procedimento Proc\_CCPP()

Se (tipo == 1) então

resultado = valor \* 1,03;

Senão

Se (tipo == 2) então

resultado = valor \* 1,05;

Senão

Mostra "Tipo desconhecido, insira 1 para poupança ou 2 para renda fixa";

Fim-se

Fim-se

Mostra resultado;

Segue

**30.**

**31.**

Algorítmo Lot01\_Rep31

Declarar

x, resultado:Int;

Inicio

Para(x=11; x<150; x++) Faça

resultado = x^2;

Mostra resultado;

Fim-Para

Fim

Debug

x | 11 | 12 | 13 | ... | 148 | 149 | 150

resultado | 121 | 144 | 169 | ... | 21904 | 22201 |

Saída

121

144

169

...

21904

22201

Algorítmo Funcao\_Quadrado(num:int)

Declarar

quadrado:int;

Inicio

quadrado = num ^ 2;

Retorna(quadrado);

Fim-Segue

31 - Mod(Função)

Algorítmo Lot01\_Rep31

Declarar

x:Int;

Inicio

Para(x=11; x<150; x++) Faça

Mostra Funcao\_Quadrado(x);

Fim-Para

Fim

Debug

x | 11 | 12 | ... | 148 | 149 | 150

Saída

121

144

...

21904

22201

**32.** **Receba um número inteiro. Calcule e mostre o seu fatorial.**

Algorítmo Lot01\_Rep32

Declara

x, y:Int;

Inicio

Ler x;

y = x;

Enquanto (x>1)

y = y \* (x-1);

x--;

Fim-Enquanto

Mostra y;

Fim

Debug

x | 0 | 1 | 5 | 4 | 3 | 2 | 1

y | 0 | 1 | 5 | 20 | 60 | 120 |

Massa de Dados

0

1

5

Saída

0

1

120

32 - Mod(Função)

Algorítmo Lot01\_Rep32

Declara

x:Int;

Inicio

Ler x;

Mostra Funcao\_Fatorial(x);

Fim

Algorítmo Funcao\_Fatorial(valor:int)

Declarar

Fat:int

Inicio

Fat = 1;

Para(i = valor; i>1;i--) Faca

Fat = Fat \* i;

Fim-Para

Retorna(Fat);

Fim-Segue

**33.** **Receba um número. Calcule e mostre a série 1 + 1/2 + 1/3 + ... + 1/N.**

Algorítmo Lot01\_Rep33

Declarar

n:Int

soma:Num;

Inicio

Ler n;

Enquanto (n>0)

soma = soma + 1/n;

n--;

Fim-Enquanto

Mostra soma;

Fim

Debug

n | 5 | 4 | 3 | 2 | 1 | 0

soma | 0.2 | 0.45 | 0.78 | 1.28 | 2.28 |

Massa de Dados

5

Saída

(2.28)

33 - Mod(Função)

Algorítmo Lot01\_Rep33

Declarar

n:Int;

Inicio

Ler n;

Mostra Funcao\_SomaInversa(n);

Fim

Algoritmo Funcao\_SomaInversa(num:num)

Declarar

soma:num;

Inicio

Enquanto (num>0)

soma = soma + 1/num;

num--;

Fim-Enquanto

Retorna soma;

Fim-Segue

**34.** **Receba um número. Calcule e mostre os resultados da tabuada desse número.**

Algorítmo Lot01\_Rep34

Declarar

x, tabuada:Int;

Inicio

Ler x;

Para (i = 0; i<=10; i++) Faça

tabuada = x \* i;

Mostra tabuada;

Fim-Para

Fim

Debug

x | 5

i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11

Massa de Dados

5

Saída

0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50

34 - Mod (Função)

Algorítmo Lot01\_Rep34

Declarar

x:Int;

Inicio

Ler x;

Funcao\_Tabuada(x);

Fim

Algoritmo Funcao\_Tabuada(valor:int)

Declarar

Inicio

Para(i = 0; i<=10; i++)

Mostra valor \* i;

Fim-Para

Retorna 0;

Fim-Segue

**35. Receba 2 números inteiros, verifique qual o maior entre eles. Calcule e mostre o resultado da somatória dos números ímpares entre esses valores.**

Algorítmo Lot01\_Rep35

Declarar

x, y, maior, menor, resto, somatoria:Int;

Inicio

Ler x;

Ler y;

somatoria = 0;

Se(x>=y) então

maior = x;

menor = y;

Senao

maior = y;

menor = x;

Fim-Se

Para(i = maior-1; i > menor; i--) Faça

resto = i%2;

Se(resto =! 0) então

somatoria = somatoria + i;

Fim-Se

Fim-Para

Mostra somatoria;

Fim

Debug

x | 5 | 8 | 4 |

y | 5 | 4 | 8 |

maior | 5 | 8 | 8 |

menor | 5 | 4 | 4 |

i | | 8 | 7 | 6 | 5 | 4 |

resto | | 0 | 1 | 0 | 1 | |

somatoria | 0 | 0 | 7 | 12 | | |

Massa de Dados

5, 5

8, 4

4, 8 //O terceiro teste leva ao mesmo resultado por ser a mesma entrada de dados, então essa parte é apenas para testar o senao.

Saída

0

12

12

35 - Mod (Função)

Algorítmo Lot01\_Rep35

Declarar

x, y:Int;

Inicio

Ler x;

Ler y;

Mostra Funcao\_SomatoriaImpares(x, y);

Fim

Algorítmo Funcao\_Maior(valor1:int, valor2:int)

Declarar

Inicio

Se(valor1 > valor2)

Return Valor1;

Senão

Return Valor2;

Fim-Se

Fim-Segue

Algorítmo Funcao\_Menor(valor1:int, valor2:int)

Declarar

Inicio

Se(valor1 < valor2)

Return Valor1;

Senão

Return Valor2;

Fim-Se

Fim-Segue

Algoritmo Funcao\_SomatoriaImpares(valor1:int, valor2:int)

Declarar

resto, somatoria:int;

Inicio

maior = Funcao\_Maior(valor1, valor2);

menor = Funcao\_Menor(valor1, valor2);

Para (i=maior-1; i>menor; i--) Faca

resto = i%2;

Se(resto != 0) então

somatoria = somatoria + i;

Fim-Se

Fim-Para

Retorna somatoria;

Fim-Segue

**36.** **Receba um número N. Calcule e mostre a série 1 + 1/1! + 1/2! + ... + 1/N!**

Algorítmo Lot01\_Rep36

Declarar

n, x:Int;

fatorial, serie:Num;

Inicio

Ler n;

n = n + 1;

x = 1;

fatorial = 1;

serie = 1;

enquanto(x<=n)

fatorial = fatorial \* x;

x++;

Mostra serie;

serie = serie + 1/fatorial;

Fim-enquanto

Fim

Debug

n | 5 | 6 | | | |

fatorial | 1 | 1 | 2 | 6 | 24 | 120 | 720

x | 1 | 2 | 3 | 4 | 5 | 6 | 7

serie | 1 | 2 | 2.50 | 2.66 | 2.70 | 2.71 | 2.71

Massa de Dados

5

Saída

1

2

(2.50)

(2.66)

(2.70)

(2.71)

36 - Mod (Função)

Algorítmo Lot01\_Rep36

Declarar

n:Int;

Inicio

Ler n;

Mostra Funcao\_InversoFatorial(n);

Fim

Algorítmo Funcao\_InversoFatorial(n:int)

Declarar

x:int;

serie:num;

Inicio

n = n + 1;

x = 1;

serie = 1;

enquanto(x<=n)

Mostra serie;

serie = serie + 1/Funcao\_Fatorial(x); //A função é do exercício 32

x++;

Fim-enquanto

Retorna serie;

Fim-Segue

**37.** **Receba um número inteiro. Calcule e mostre a série de Fibonacci até o seu N’nésimo termo.**

Algorítmo Lot01\_Rep37

Declarar

n, valor1, valor2, troca :num;

Inicio

Ler n;

valor2 = 1;

Para(i=0;i<=n;i++) Faca

mostra valor1;

troca = valor1;

valor1 = valor2;

valor2 = troca;

valor1 = valor1 + valor2;

Fim-Para

Fim

Debug

n | 5 |

valor1 | 0 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 3 | 2 | 5 | 3 | 8

valor2 | 1 | 0 | 1 | 1 | 2 | 3 | 5

troca | 0 | 1 | 1 | 2 | 3 | 5

i | 0 | 1 | 2 | 3 | 4 | 5 | 6

Massa de Dados

6

Saída

0

1

1

2

3

5

37 - Mod (Função)

Declarar

n:int;

Inicio

Ler n;

Funcao\_Fibonacci(n);

Fim

Algoritmo Funcao\_Fibonacci(n:int)

Declarar

valor1, valor2, troca :num;

Inicio

valor2 = 1;

Para(i=0;i<=n;i++) Faca

mostra valor1;

troca = valor1;

valor1 = valor2;

valor2 = troca;

valor1 = valor1 + valor2;

Fim-Para

Retorna 0;

Fim-Segue

**38.** **Receba 100 números inteiros reais. Verifique e mostre o maior e o menor valor. Obs.: somente valores positivos.**

Algorítmo Lot01\_Rep38

Declarar

menor, maior, n : int;

Inicio

Para (i=0;i<=100;i++) Faca

Leia n;

Se (n < 0) Então

Mostra "Erro! Insira somente valores positivos!";

i--;

Senão

Se (n < menor) Então

menor = n;

Fim-Se

Se (n > maior) Então

maior = n;

Fim-Se

Fim-Se

Fim-Para

Mostra "Menor: " + menor;

Mostra "Maior: " + maior;

Fim

// feito com 3 entradas apenas

Entrada: 0 | 1 | 2 |

Saida: 0 | 2 |

menor: 0 |

maior: 1 | 2 |

n: 0 | 1 | 2 |

// feito com valor negativo

Entrada: -1 |

Saida: Erro insira somente valores positivos

menor:

maior:

n: -1 |

38 - Mod (Função)

Declarar

menor, maior, n : int;

Inicio

Para (i=0;i<=100; i++) Faca

Leia n;

Se (n < 0) Então

Mostra "Erro! Insira somente valores positivos!";

i--;

Senão

menor = Funcao\_Menor(n, menor); //Função do exercício 35

maior = Funcao\_Maior(n, maior); //Funçao do exercício 35

Fim-Se

Fim-Para

Mostra "Menor: " + menor;

Mostra "Maior: " + maior;

Fim

**39.** **Calcule a quantidade de grãos contidos em um tabuleiro de xadrez onde:**

Algorítmo Lot01\_Rep39

Declarar

casa, qnt:Int;

Inicio

casa = 64

Para (i = 0; i < casa; i++) Faca

qnt = 2 ^ i;

Fim-Para

Mostra qnt;

Fim

Debug

casa | 64 | |

i | 0 | 1 | 2 | 3 | 4 | ... | 63 | 64

qnt | 1 | 2 | 4 | 8 | 16 | ... | 9,223,372,036,854,775,808 |

Saída

9,223,372,036,854,775,808

39 - Mod (Função)

Algorítmo Lot01\_Rep39

Declarar

casa, qnt:Int;

Inicio

casa = 64

Para (i = 0; i < casa; i++) Faca

qnt = Funcao\_ExpoenteBase2(i);

Fim-Para

Mostra qnt;

Fim

Algoritmo Funcao\_ExpoenteBase2(expoente:int)

Declarar

Valor

Inicio

valor = 2 ^ expoente;

Retorna expoente;

Fim-Segue

**40.** **Receba 2 números inteiros. Verifique e mostre todos os números primos existentes entre eles.**

Declarar

valor1, valor2, p, r, t, i :int;

INICIO

Ler valor1;

Ler valor2;

Enquanto (valor1 < valor2)

p = 1;

Para (i=2; i<valor2; i++) Faca

r = valor2 / i;

t = R - Int(r);

Se (t==0) Então

p = 0;

Fim-Se

Fim-Para

Se (p == 1) Então

Mostra valor2;

Fim-Se

Fim-Enquanto

Fim

Valor2: 6

R: 3

T: 0

I: 2

Valor2: 6

R: 2

T: 0

I: 3

Valor2: 6

R: 1.5

T: 0.5

I: 4

Valor2: 6

R: 1.2

T: 0.19999999999999996

I: 5

Valor2: 5

R: 2.5

T: 0.5

I: 2

Valor2: 5

R: 1.6666666666666667

T: 0.6666666666666667

I: 3

Valor2: 5

R: 1.25

T: 0.25

I: 4

VALOR2: 5

Valor2: 4

R: 2

T: 0

I: 2

Valor2: 4

R: 1.3333333333333333

T: 0.33333333333333326

I: 3

Valor2: 3

R: 1.5

T: 0.5

I: 2

VALOR2: 3

**41.** **Mostre todas as possibilidades de 2 dados de forma que a soma tenha como resultado 7.**

Algorítmo Lot01\_Rep41

Declarar

x, y :int;

Inicio

Para (x=1;x<=6;x++) Faca

Para (y=1;y<=6;y++) Faca

Se (x + y == 7) Então

Mostra x + " + " + y + " = 7";

Fim-Se

Fim-Para

Fim-Para

Fim

Debug

x | 1 | 2 | 3 | 4 | 5 | 6 | 7

y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7

Saída

1 + " + " + 6 + " =7"

2 + " + " + 5 + " =7"

3 + " + " + 4 + " =7"

4 + " + " + 3 + " =7"

5 + " + " + 2 + " =7"

6 + " + " + 1 + " =7"

41. Mod(Função)

Declarar

x, y :int;

Inicio

x=6;

y=6;

Funcao\_DadosSoma7(x,y);

Fim

Algoritmo Funcao\_DadosSoma7(x:int, y:int)

Declarar

Inicio

Para (i=1;i<=x;i++) Faca

Para (k=1;k<=y;k++) Faca

Se (i + k == 7) Então

Mostra i + " + " + k + " = 7";

Fim-Se

Fim-Para

Fim-Para

Retorna 0;

Fim-Segue

**42. Calcule e mostre a série 1 + 2/3 + 3/5 + ... + 50/99**

Algorítmo Lot01\_Rep41

Declarar

x, y, resultado :num;

Inicio

x = 1;

Enquanto (x <= 50) Faca

y = (x\*2)-1;

resultado = resultado + (x/y);

x++;

Mostra resultado;

Fim-Enquanto

Fim

Debug

x | 1 | 2 | 3 | ...

y | 1 | 3 | 5 | ...

resultado | 1 | 1.66 | 2.26 | ...

Saída

1

1.66

2.26

...

42 - Mod(Função)

Declarar

x:int;

Inicio

x=50;

Funcao\_Serie(x);

Fim

Algoritmo Funcao\_Serie(posicao:int)

Declarar

x :int;

Inicio

x = 1;

Enquanto (x <= posicao) Faca

y = (x\*2)-1;

resultado = resultado + (x/y);

x++;

Mostra resultado;

Fim-Enquanto

Retorna 0;

Fim

**43. Calcule e mostre quantos anos serão necessários para que Ana seja maior que Maria sabendo que Ana tem 1,10 m e cresce 3 cm ao ano e Maria tem 1,5 m e cresce 2 cm ao ano.**

Algorítmo Lot01\_Rep43

Declarar

ana, maria, ano :int;

Inicio

ana = 110;

maria = 150;

enquanto (ana <= maria) faca

ana = maria + 3;

maria = ana + 2;

ano++;

Fim-Enquanto

Mostra ano;

Fim

Debug

ana | 110 | 113 | 116 | 119 | 122 | 125 | ... | 230 | 233

maria | 150 | 152 | 154 | 156 | 158 | 160 | ... | 230 | 233

ano | | 1 | 2 | 3 | 4 | 5 | ... | 40 | 41

Saída

41

43 - Mod (Função)

Declarar

ana, maria, ano :int;

Inicio

ana = 110;

maria = 150;

Mostra Funcao\_CalculoAltura(ana, maria);

Fim

Algoritmo Funcao\_CalculoAltura(ana:int, maria:int)

Declarar

Inicio

Enquanto (ana <= maria) Faca

ana = maria + 3;

maria = ana + 2;

ano++;

Fim-Enquanto

Retorna ano;

Fim-Segue

**44.** **Receba o número da base e do expoente. Calcule e mostre o valor da potência.**

**Algoritmo potencia**

Declarar

base, expoente, resultado :int;

Inicio

Ler base;

Ler expoente;

resultado = base;

Para (expoente; expoente > 1; expoente--) Faca

resultado = (resultado \* base);

Fim-Para

Mostra resultado;

Fim

**Teste de mesa**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Base** | 2 |  |  |  |
| **Expoente** | 4 | 3 | 2 | 1 |
| **Resultado** | 2 | 4 | 8 | 16 |

**Massa de Dados Saída**

2 16

4

**Mod (Função)**

Declarar

base, expoente:int;

Inicio

Ler base;

Ler expoente;

Mostra Funcao\_Potenciacao(base, expoente);

Fim

**Algoritmo Funcao\_Potenciacao(base:int, expoente:int)**

Declarar

resultado:int;

Inicio

resultado = base;

Para(i = exponte; i > 1; i--) Faca

resultado = resultado \* base;

Fim-Para

Retorna resultado;

Fim

**45. Calcule e mostre a série 1 – 2/4 + 3/9 – 4/16 + 5/25 + ... + 15/225**

**Algoritmo Lot01\_Rep45**

Declarar

x, y, serie :int;

Inicio

x = 1;

Enquanto (x <= 15) Faca

y = x ^ 2

Se (x%2 == 0) Então

serie = serie - (x / y);

Senão

serie = serie + (x / y);

Fim-se

Mostra serie;

x++;

Fim-Enquanto

Fim

**Teste de mesa**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **X** | 1 | 2 | 3 | 4 | ... | 15 |
| **Y** | 1 | 4 | 9 | 16 | ... | 225 |
| **Serie** | 1 | 0.5 | 0.83 | 0.78 | ... | 0.73 |

**Massa de dados Saída**

1 2 3 4 ... 15 1

1 4 9 16 ... 225 0.5

0.83

0.78

...

0.73

**Mod (Função)**

Declarar

x:int

Inicio

x = 15

Funcao\_Serie(x);

Fim

**Algoritmo Funcao\_Serie(posicao:int)**

Declarar

x, y:int

serie:num

Inicio

x = 1;

Enquanto (x <= posicao) Faca

y = x ^ 2

Se (x%2 == 0) Então

serie = serie - (x / y);

Senão

serie = serie + (x / y);

Fim-Se

Mostra serie;

x++;

Fim-Enquanto

Return 0;

Fim