

Linguagem de Programação

Estruturas de Repetição

Profa. Eliane Oliveira Santiago

Estruturas de Repetição

Quando desenvolvemos um programa estruturado, muitas vezes, precisamos repetir parte do algoritmo, um comando ou um conjunto de comandos.

Podemos fazer isso repetindo o bloco quantas vezes forem necessárias, mas isso é inviável.

Para solucionar esse problema, podemos usar as estruturas de repetição.

No entanto, precisamos saber controlar quantas vezes o algoritmo deve repetir determinado bloco de programação/algoritmo.

Para isso, estudaremos o fluxo de controle chamado estrutura de repetição.

Estrutura de Repetição

Uma estrutura de repetição é um fluxo de controle utilizado para decidir quantas vezes determinado conjunto de comandos se repetirá dentro do algoritmo.

Uma estrutura de repetição determina qual conjunto de comandos ou bloco será executado após uma condição ser avaliada.

Essa condição é representada por expressões lógicas e relacionais que podem ou não ser satisfeitas, isto é, podem retornar o valor verdadeiro ou falso. Enquanto essa condição estiver retornando verdadeiro, o conjunto de comandos será executado, parando somente quando o resultado da avaliação da condição for falso.

Numa estrutura de repetição, o número de repetições pode ser indeterminado, mas necessariamente finito.

Laço *para* ↔ laço *for*

para <variavel> de <valor_inicial> até <valor_final> passo 1 faça

//bloco

fimpara

for(inicialização; teste_condicional; incremento){

//bloco

}

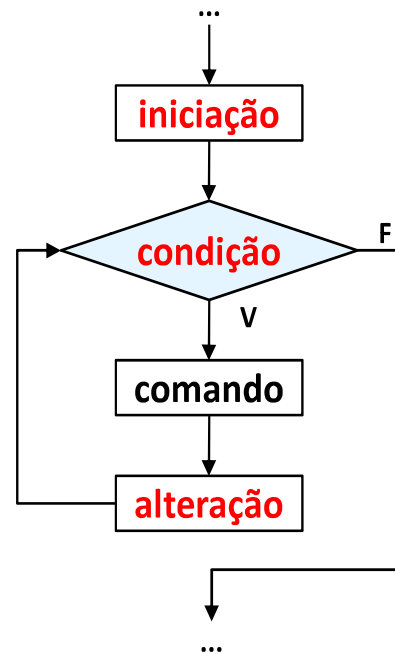
Estrutura de Repetição para um número determinado de vezes

```
for (iniciação; condição; alteração) comando;
```

Exemplo 5. O comando `for`

Exibir uma contagem progressiva de 1 até 9.

```
#include <stdio.h>
int main(void) {
    int i;
    for(i=1; i<=9; i++)
        printf("%d\n", i);
    return 0;
}
```



Exercício 1.

Dado um número n (entre 1 e 10), exiba a sua tabuada.



Exercício 1.

Dado um número n (entre 1 e 10), exiba a sua tabuada.

```
#include <stdio.h>

int main(void) {
    int n;
    printf("Numero? ");
    scanf("%d", &n);
    for(int i=1; i<=10; i++)
        printf("%d x %2d = %3d\n", n, i, n*i);
    return 0;
}
```

Exercício 2.

Dado um número positivo n , exiba uma contagem regressiva de n até 0.



Intervalo

Retornaremos às 20h50

Exercício 2.

Dado um número positivo n , exiba uma contagem regressiva de n até 0.

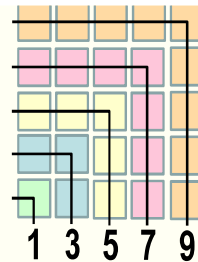
```
#include <stdio.h>

int main()
{
    int n;
    printf("Digite um valor para n: ");
    scanf("%d", &n);
    for(int i=n; i>=0; i--){
        printf("Números %d\n", i);
    }
    return 0;
}
```

Exercício 3. Quadrados perfeitos

O **quadrado** de um número natural **n** é igual à **soma** dos **n** primeiros ímpares consecutivos. Com base nessa ideia, crie um programa que, dado um número natural **n**, calcula e exibe o quadrado de **n**.

Por exemplo: $5^2 = 1 + 3 + 5 + 7 + 9 = 25$



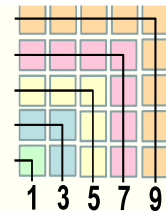
Exercício 3. Quadrados perfeitos

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

```
int main()
{
    int n, soma=0;
    printf("Digite um valor para n: ");
    scanf("%d", &n);
    for(int i=1; i<=2*n; i+=2){
        soma += i;
    }
    printf("Números %d\n", i);
    return 0;
}
```

O **quadrado** de um número natural n é igual à **soma** dos n primeiros ímpares consecutivos. Com base nessa ideia, crie um programa que, dado um número natural n , calcule e exiba o quadrado de n .

Por exemplo: $5^2 = 1 + 3 + 5 + 7 + 9 = 25$



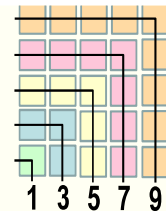
Exercício 3. Quadrados perfeitos

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

```
int main()
{
    int n, soma=0;
    printf("Digite um valor para n: ");
    scanf("%d", &n);
    for(int i=1; i<=2*n; i+=2){
        soma += i;
    }
    printf("Números %d\n", i);
    return 0;
}
```

O **quadrado** de um número natural n é igual à **soma** dos n primeiros ímpares consecutivos. Com base nessa ideia, crie um programa que, dado um número natural n , calcula e exibe o quadrado de n .

Por exemplo: $5^2 = 1 + 3 + 5 + 7 + 9 = 25$



Exercício 4. Fatorial

O **fatorial** de um número natural positivo **n** é igual ao **produto** dos **n** primeiros naturais positivos (por definição, o fatorial de **0** é **1**). Dado um número natural **n**, calcule e exiba o seu fatorial.

□ $0! = 1$

□ $5! = 5.4.3.2.1$

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

```
int main()
```

```
{
```

```
    int n, fatorial = 1;
```

```
    printf("Numero? ");
```

```
    scanf("%d", &n);
```

```
    for(int i = n; i > 1; i--)
```

```
        fatorial *= i;
```

```
    printf("\n Fatorial de %d é %d ", n, fatorial);
```

```
    return 0;
```

```
}
```

Exercício 5. Terminal

O **termial** de um número natural positivo **n** é igual à **soma** dos **n** primeiros naturais positivos (por definição, o termial de **0** é **0**). Dado um número natural **n**, calcule e exiba o seu termial.

```
#include <stdio.h>

int main()
{
    int n, terminal = 0;

    printf("Numero? ");
    scanf("%d", &n);

    for(int i = 0; i<=n; i++)
        terminal +=i;

    printf("\n O número terminal de %d é %d ",n, terminal);
    return 0;
}
```

For encadeado

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

```
int main()
```

```
{
```

```
    int i, j;
```

```
    for(int i = 1; i<=10; i++)
```

```
        for(int j = 1; j<=10; j++)
```

```
            printf("\n %d * %d = %d ",i,j, i*j);
```

```
    return 0;
```

```
}
```

For encadeado

```
#include <stdio.h>

int main()
{
    int M[5][5];
    for(int i=0; i<5; i++)
        for(int j=0; j<5; j++)
            M[i][j]=i+j;

    return 0;
}
```


For encadeado

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

```
int main()
```

```
{
```

```
    int M[5][5];
```

```
    for(int i=0; i<5; i++)
```

```
        for(int j=0; j<5; j++)
```

```
            M[i][j]=i+j;
```

```
    int lin, col;
```

```
    for(lin=0, col=0; lin<5, col<5; lin++, col++)
```

```
        printf("\nM[%d][%d] = %d", lin, col, M[lin][col]);
```

```
    return 0;
```

```
}
```

Repetição com teste no início

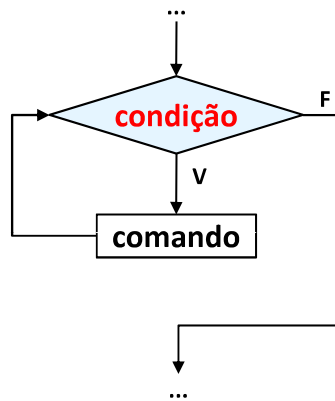
Repete um comando, um número indeterminado de vezes (zero ou mais).

```
while(condição){  
    //instruções  
}
```

○ comando while

Dado um número natural positivo, exiba os seus dígitos.

```
#include <stdio.h>
int main(void) {
    int n;
    printf("Numero? ");
    scanf("%d", &n);
    while( n>0 ) {
        printf("%d\\n", n%10);
        n /= 10;
    }
    return 0;
}
```



```
while ( condição )
    comando;
```

Estrutura de Repetição com teste no início

Enquanto (<condição>) faça

```
<inicialização da variável de controle>;  
enquanto (<condição>) faça  
    <comandos>;  
    <atualização da variável de controle>;  
fimenquanto;
```

Por exemplo:

```
x ← 0;  
enquanto (x < 3) faça  
    escreva ("O valor de x é: ", x);  
    x ← x + 1;  
fimenquanto;
```

Nesse exemplo, o valor inicial da variável x é 0.

Na primeira vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 0 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 1) são executados.

Na segunda vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 1 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 2) são executados.

Na terceira vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 2 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 3) são executados.

Na quarta vez que a condição $x < 3$ é avaliada, seu resultado é falso, pois 3 não é menor que 3, então os comandos dentro da estrutura de repetição enquanto não são executados e essa estrutura é finalizada, seguindo com o fluxo do algoritmo após esta estrutura.

O resultado da execução deste exemplo é:

O valor de x é 0

O valor de x é 1

O valor de x é 2

Estrutura de Repetição com teste no início

Enquanto (<condição>) faça

```
<inicialização da variável de controle>;  
enquanto (<condição>) faça  
    <comandos>;  
    <atualização da variável de controle>;  
fimenquanto;
```

Por exemplo:

```
x ← 0;  
enquanto (x < 3) faça  
    escreva ("O valor de x é: ", x);  
    x ← x + 1;  
fimenquanto;
```

Nesse exemplo, o valor inicial da variável x é 0.

Na primeira vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 0 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 1) são executados.

Na segunda vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 1 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 2) são executados.

Na terceira vez que a condição $x < 3$ é avaliada, seu resultado é verdadeiro, pois 2 é menor que 3, então o comando escreva(...); e a atualização de x (x passa a ser 3) são executados.

Na quarta vez que a condição $x < 3$ é avaliada, seu resultado é falso, pois 3 não é menor que 3, então os comandos dentro da estrutura de repetição enquanto não são executados e essa estrutura é finalizada, seguindo com o fluxo do algoritmo após esta estrutura.

O resultado da execução deste exemplo é:

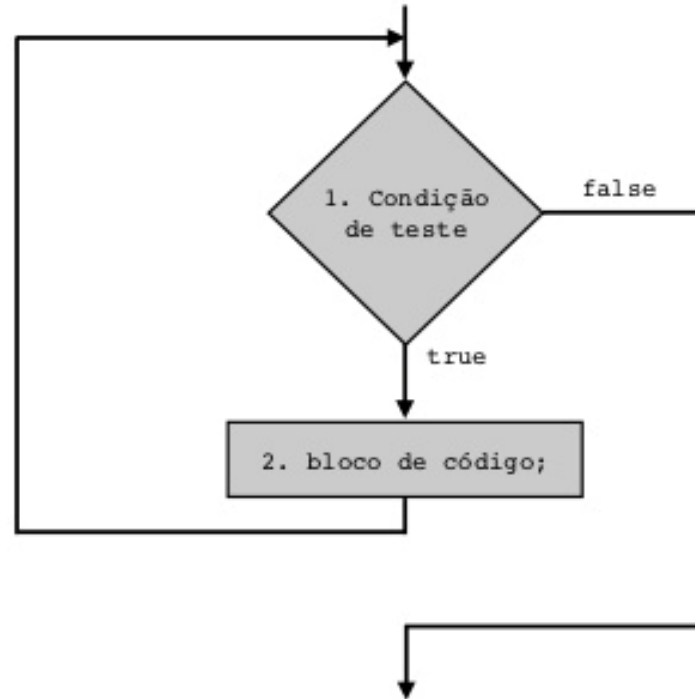
O valor de x é 0

O valor de x é 1

O valor de x é 2

Estrutura de Repetição com teste no início

Enquanto (<condição>) faça



Repetição com teste no final

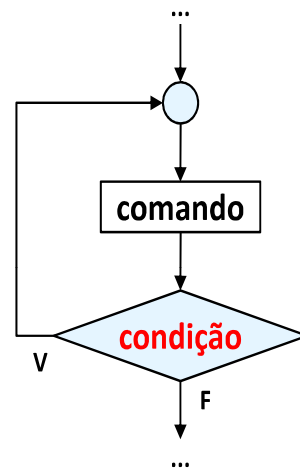
Repete um comando um número indeterminado de vezes (uma ou mais vezes).

```
do{  
    //instruções  
}while(condição)
```

○ comando do... while

Exibir a soma de uma sequência de números terminada com 0.

```
#include <stdio.h>
int main(void) {
    int s=0, n;
    do {
        printf("Numero? ");
        scanf("%d",&n);
        s += n;
    } while( n!=0 );
    printf("Soma = %d\n", s);
    return 0;
}
```



```
do {
    comando;
} while( condição );
```


O comando do... while

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    srand(time(NULL));
    int c, n = rand()%7+1;

    do{
        printf("chute entre 1 e 7: ");
        scanf("%d", &c);

        if(c>n) puts("Muito alto!");
        else if(c<n) puts("Muito baixo");
        else puts("Voce acertou");
    }while(n!=c);
    return 0;
}
```

Estrutura de Repetição com teste no fim

Faça enquanto (<condição>)

```
<inicialização da variável de controle>;  
enquanto (<condição>) faça  
    <comandos>;  
    <atualização da variável de controle>;  
fimenquanto;
```

Por exemplo:

```
x ← 0;
```

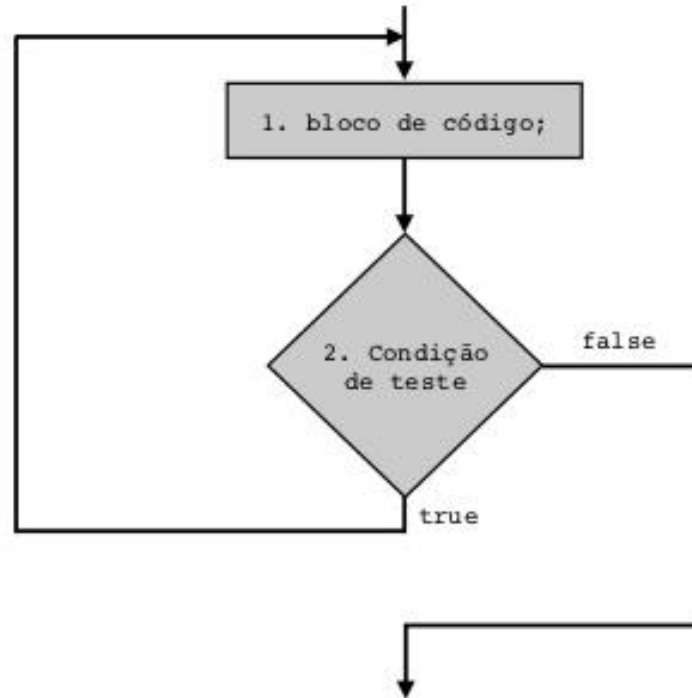
faça

```
    escreva ("O valor de x é: ", x);
```

```
    x ← x + 1;
```

enquanto(x<3)

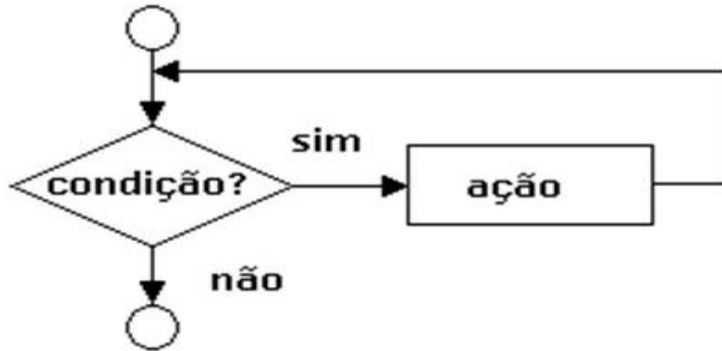
fimenquanto;



Estruturas de Repetição

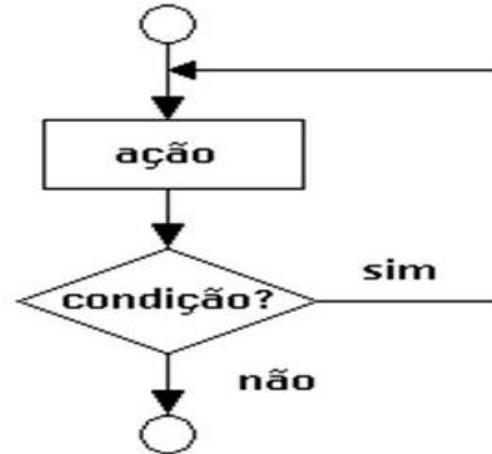
Estrutura de repetição com teste no início

```
enquanto (<condição>) faça  
    //bloco de instruções  
fim_enquanto
```



Estrutura de repetição com teste no fim

```
faça  
    //bloco de instruções  
enquanto (<condição>)
```



Encadeamento

Na estrutura de repetição enquanto, utilizaremos as palavras enquanto e faça que representam as palavras principais dessa estrutura e a palavra fim_enquanto; para determinar o fim do bloco de execução dessa estrutura. A estrutura de repetição enquanto encadeada para pseudocódigo segue a seguinte regra sintática:

Solução do Exercício 1.

Tabuada de um número n


```
#include <stdio.h>

int main()
{
    int n, j, r;
    printf("N = ");
    scanf("%d", &n);
    i = 0;
    while (i<=10){
        r = n * i
        printf("\n %d * %d = %d", n, i, r)
        i++;
    }
}
```

Exercício 2. Elaborar um programa que mostre os resultados da tabuada do 1 ao 10.

```
programa Tabuada_do_1_ao_10
var
    cont, n, res: inteiro
início
    cont = 0
    n = 0
    enquanto (cont<=10) faça
        enquanto(n<=10) faça
            res ← cont * n
            escreva(cont, " * ", n, " = ", res)
            n ← n + 1
        fim_enquanto.
        cont ← cont +1
        n ← 0
    fim_enquanto
Fim.
```

c) Elaborar um programa que leia 2 números inteiros (x e y) e implemente uma Calculadora Quebrada que efetue as 4 operações, mas sem usar a tecla $/$ ou $*$.



c) Elaborar um programa que leia 2 números inteiros (x e y) e implemente uma Calculadora Quebrada que efetue as 4 operações, mas sem usar a tecla / ou *.

```
1. programa CalculadoraQuebrada
2. var
3.   i, resto, r, x, y : int
4.   operador: caracter

5. inicio
6.   escreva("Calculadora Quebrada")

7.   leia(x,y)

8.   escreva("[+] Somar")
9.   escreva("[-] Subtrair")
10.  escreva("[*] Multiplicar")
11.  escreva("[/] Dividir")

12.  escreva("Escolha uma opção de operador")
```

```
13.  leia(operador)
14.  escolha(operador)
15.    caso "+":
16.      r = x + y
17.      escreva("A soma de ", x , " e ", y, "eh igual a ", r)
18.    caso "-":
19.      r = x - y
20.      escreva("A diferença entre ", x , " e ", y, "eh igual a ", r)
```

```
21. caso "*":
22.     i = 1
23.     r = 0
24.     enquanto(i <= y) faca
25.       r = r + x
26.       i = i+1
27.     fim_enquanto
28.     escreva(x, " multiplicado por ", y, "eh igual a ", r)
```


c) Elaborar um programa que leia 2 números inteiros (x e y) e implementar uma Calculadora Quebrada que efetua as 4 operações, mas sem usar a tecla / ou *

```
29. caso "/":
30.     i=0
31.     r = x
32.     enquanto(x<=y) faça
33.         i = i + 1  //i guardará o quociente da divisão
34.         x = x - y  //x guardará o resto da divisão
35.     fim_enquanto

36.     caso_contrario:
37.         escreva("operador não reconhecido")
38.     fim_enquanto
39. fim
```

Exercícios

1. Desenvolva um algoritmo que calcule e mostre o quadrado dos números inteiros compreendidos entre 10 e 150.
2. Desenvolva um algoritmo que receba um número inteiro, calcule e mostre o seu fatorial.
3. Desenvolva um algoritmo que receba um número N , calcule e mostre o valor da seguinte série:
Série = $1 + 1/2 + 1/3 + \dots + 1/N$.
4. Desenvolva um algoritmo que receba um número, calcule e mostre os resultados da tabuada desse número.
5. Desenvolva um algoritmo que receba dois números inteiros, verifique qual é o maior entre eles, calcule e mostre o resultado da somatória dos números ímpares compreendidos entre esses dois números.

Exercícios (2)

6. Desenvolva um algoritmo que receba um número N , calcule e mostre o valor da seguinte série:

$$\text{Série} = 1 + 1/1! + 1/2! + 1/3! + \dots + 1/N!$$

7. Desenvolva um algoritmo que receba um número inteiro N , calcule e mostre a série de Fibonacci até o seu N -ésimo termo. A série de Fibonacci é dada pela seguinte seqüência: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ... etc.

8. Desenvolva um algoritmo que receba 100 números reais, verifique e mostre o maior e o menor número recebido.

9. Desenvolva um algoritmo que calcule o número de grãos de trigo dispostos num tabuleiro de xadrez e que segue a seguinte regra: no primeiro quadro, colocar um grão de trigo; no segundo quadro, colocar o dobro de grãos de trigo do primeiro quadro; e, para todos os quadros subsequentes, colocar o dobro de grãos de trigo do quadro anterior. Um tabuleiro de xadrez tem 64 quadros.

Exercícios (3)

10. Desenvolva um algoritmo que receba um número x calcule e mostre um número inteiro que mais se aproxima da raiz quadrada desse número x .
11. Desenvolva um algoritmo que receba um número inteiro, verifique e mostre se esse número é primo ou não.
12. Desenvolva um algoritmo que receba dois números inteiros, verifique e mostre todos os números primos existentes entre os dois números recebidos.
13. Desenvolva um algoritmo que mostre todas as possibilidades de lançamento de dois dados, de forma que tenhamos o valor 7 como resultado da soma dos valores de cada dado.
14. Desenvolva um algoritmo que calcule e mostre o valor da seguinte série:
Série = $1 + 2/3 + 3/5 + 4/7 + \dots + 50/99$.
15. Desenvolva um algoritmo que receba 15 números inteiros, calcule e mostre a somatória do fatorial de cada número recebido.

Exercícios (4)

16. Desenvolva um algoritmo que recebe 30 números reais, calcula e mostra a somatória e a média dos números recebidos.
17. Desenvolva um algoritmo que receba dois números inteiros, verifique qual o maior entre eles, calcule e mostre o quociente e o resto da divisão do maior número pelo menor. Em hipótese nenhuma utilizar os operadores div e mod.
18. Desenvolva um algoritmo que receba o nome e a idade de 50 pessoas, verifique e mostre o nome e a idade da pessoa mais velha e da pessoa mais nova.
19. Desenvolva um algoritmo que calcule e mostre quantos anos serão necessários para que Florentina seja maior que Clarisbela, considerando que Clarisbela tem 1,50 metros e cresce 2 centímetros por ano e Florentina tem 1,10 metros e cresce 3 centímetros por ano.
20. Desenvolva um algoritmo que calcule e mostre o valor da somatória dos trinta primeiros termos da seguinte série:
- $$\text{Série} = 5/1000 - 10/980 + 5/960 - 10/940 + \dots$$

Exercícios (5)

21. Desenvolva um algoritmo que receba o salário de Clarisbela, calcule e mostre o salário de Florentina que é um terço do salário de Clarisbela. Além disso, calcule e mostre quantos meses são necessários para que o dinheiro aplicado de Florentina seja igual ou maior que o dinheiro aplicado de Clarisbela, considerando que Clarisbela aplicou todo o seu salário na poupança, que rende 2% ao mês, e que Florentina aplicou todo o seu salário no fundo de renda fixa, que rende 5% ao mês.

22. Desenvolva um algoritmo que receba dois valores inteiros, x e y , calcule e mostre a potência xy , sem utilizar o operador pot.

23. Desenvolva um algoritmo que receba o sexo, a idade e a experiência no trabalho (s/n) de 50 candidatos a uma vaga de uma empresa, calcule e mostre: o número de candidatos do sexo masculino; o número de candidatos do sexo feminino; a idade média dos candidatos que já têm experiência no trabalho; a porcentagem de homens com mais de 50 anos do total de homens; o número de mulheres com idade inferior a 30 e com experiência no trabalho; e a menor e a maior idade entre as mulheres que já têm experiência no trabalho.

Exercícios (6)

24. Desenvolva um algoritmo que receba 50 números reais, calcule e mostre: a soma dos números digitados; a média dos números digitados; o maior número digitado; o menor número digitado; o dobro dos números digitados; o cubo dos números digitados; a porcentagem dos números ímpares digitados.

25. Desenvolva um algoritmo que calcule e mostre o valor da seguinte série:

$$\text{Série} = 1/2 - 2/4 + 3/6 - 4/8 + 5/10 - 6/12 + \dots - 50/100.$$

26. Desenvolva um algoritmo que receba um número inteiro N, mostre um menu de opções: calcule e mostre a soma do fatorial dos números entre 1 e N; calcule e mostre a série de Fibonacci até o seu N-ésimo termo; calcule e mostre o produto dos números entre 1 e N; e calcule e mostre a soma dos números pares entre 1 e N.

Exercícios (7)

27. Desenvolva um algoritmo que mostre um menu de opções: receba 20 números reais, calcule e mostre a somatória desses números; receba 15 números reais, calcule e mostre o produto dos números ímpares desses números; e receba 10 números inteiros, calcule e mostre quais desses números são divisíveis por 5 e 7.

28. Desenvolva um algoritmo que mostre um menu de opções: receba a altura e a base de 10 triângulos, calcule e mostre a área de cada triângulo; receba a altura e a base de 15 retângulos, calcule e mostre a área de cada retângulo; receba a altura, a base maior e a base menor de 20 trapézios, calcule e mostre a área de cada trapézio; e receba o raio de 15 circunferências, calcule e mostre a área de cada circunferência.

Exercícios (8)

29. Desenvolva um algoritmo que receba dois valores inteiros, a e b , $a > b$, mostre um menu de opções que: calcule e mostre a potência a^b , sem utilizar o operador `pot`; calcule e mostre o quociente e o resto da divisão de a por b , sem usar os operadores `div` e `mod`; e calcule e mostre se os valores a e b são primos ou não.

30. Desenvolva um algoritmo que calcule e mostre o valor da seguinte série:

$$\text{Série} = 1/1 - 2/4 + 3/9 - 4/16 + 5/25 - \dots + 15/225.$$