

Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

28/08/2024

Algoritmos: Estruturas de Repetição

Aula anterior

Estruturas de Repetição

Planejando laços para
solucionar problemas

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Considere: base um valor real.

Expoente: um número inteiro, positivo ou nulo (zero).

*Obs: Não usar a biblioteca Math. A potenciação deverá ser calculada
de forma iterativa.*

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Analisando o problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Analizando o problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Requer acumulador para cada etapa do cálculo do produto

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Requer acumulador para cada etapa do cálculo do produto

`potenciacao ← potenciacao * base`

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Requer acumulador para cada etapa do cálculo do produto



potenciacao ← potenciacao * base

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Requer acumulador para cada etapa do cálculo do produto

*Número de
iterações?*



`potenciacao ← potenciacao * base`

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação:

$$2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$$

Requer acumulador para cada etapa do cálculo do produto

Expoente vezes  `potenciacao ← potenciacao * base`

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

Com qual valor o acumulador deverá ser inicializado?

Expoente vezes  `potenciacao ← potenciacao * base`

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

potenciacao $\leftarrow 1$

Expoente vezes  `potenciacao \leftarrow potenciacao * base`

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

potenciacao \leftarrow 1;

para i **de** 1 **até** expoente **passo** 1 **faça**

potenciacao \leftarrow potenciacao * base

fim_para



Calcular e escrever a potenciação.

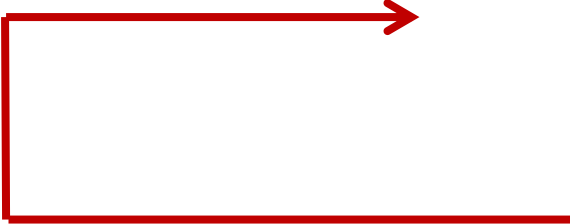
*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

Modelando a solução do problema:

Potenciação: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

potenciacao = 1;

Expoente
vezes



```
for( int i=1 ; i <= expoente ; i++ ) {  
    potenciacao = potenciacao * base ;  
}
```


Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

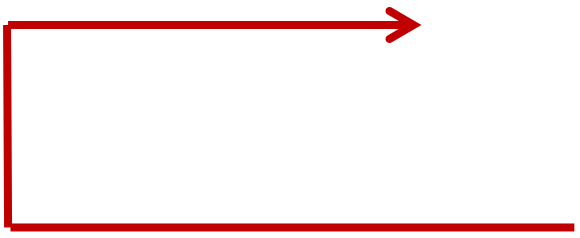
Modelando a solução do problema:

Potenciação: $2^8 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$

leia(base, expoente);

potenciacao = 1;

Expoente
vezes



```
for( int i=1 ; i <= expoente ; i++ ) {  
    potenciacao = potenciacao * base ;  
}
```

Potenciação

Codificando o algoritmo em C

*Calcular e escrever a potenciação.
A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()  
{  
    printf("Hello world!\n");  
  
    return 0;  
} // fim main()
```

*Calcular e escrever a potenciação.
A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()  
{  
    printf("Calculo da potenciacao\n");  
  
    return 0;  
} // fim main()
```

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("Calculo da potenciacao\n");

    leia( base, expoente );
    potenciacao = 1;

    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }

    return 0;
} // fim main()
```

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("\nCalculo da potenciacao\n");

    float base;
    printf("\nBase: ");
    scanf("%f", &base);

    int expoente;
    printf("\nExpoente: ");
    scanf("%i", &expoente);

    float potenciacao = 1;
    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }
    return 0;
} // fim main()
```


Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("\nCalculo da potenciacao\n");
    float base;
    printf("\nBase: ");
    scanf("%f", &base);

    int expoente;
    printf("\nExpoente: ");
    scanf("%i", &expoente);

    float potenciacao = 1;
    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }

    return 0;
} // fim main()
```

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("\nCalculo da potenciacao\n");
    float base;
    printf("\nBase: ");
    scanf("%f", &base);
    int expoente;
    printf("\nExpoente: ");
    scanf("%i", &expoente);
    float potenciacao = 1;
    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }
    printf("\n%f ^ %i = %f", base, expoente, potenciacao);
    return 0;
} // fim main()
```

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("\nCalculo da potenciacao\n");
    float base;
    printf("\nBase: ");
    scanf("%f", &base);

    int expoente;
    printf("\nExpoente: ");
    scanf("%i", &expoente);

    float potenciacao = 1;
    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }

    printf("\n%.1f ^ %i = %.1f", base, expoente, potenciacao);
    return 0;
} // fim main()
```

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
int main()
{
    printf("\nCalculo da potenciacao\n");
    float base;
    printf("\nBase: ");
    scanf("%f", &base);

    int expoente;
    printf("\nExpoente: ");
    scanf("%i", &expoente);

    float potenciacao = 1;
    for( int i=1 ; i <= expoente ; i++ ) {
        potenciacao = potenciacao * base ;
    }
    printf("\n%f ^ %i = %f", base, expoente, potenciacao);
    return 0;
} // fim main()
```

*Rejeitar a leitura de
expoente negativo*

Calcular e escrever a potenciação.

*A base e o expoente deverão ser valores lidos –
rejeitar a leitura de valores inválidos.*

```
#include <stdbool.h>
```

```
...
```

```
bool erro;
```

```
int expoente;
```

```
do {
```

```
    printf("\nExpoente: ");
```

```
    scanf("%i", &expoente);
```

```
    erro = expoente < 0;
```

```
    if(erro) printf("\nApenas valores positivos");
```

```
}while( erro );
```

Algoritmos: Estruturas de Repetição

Planejando laços para
solucionar problemas

Soluções iterativas requer laços

Vimos que há três estratégias para laços:

- ✓ Com teste no início: `while(?) { ... }`
- ✓ Com teste no final: `do { ... } while(?);`
- ✓ Com variável de controle: `for(... ; ? ; ...) { ... }`

Estratégias iterativas

Contador:

Estratégias iterativas

Acumulador:

Estratégias iterativas

Maior valor com domínio bem definido

Estratégias iterativas

Menor valor com domínio bem definido

Estratégias iterativas

Maior/menor valor sem domínio

Estratégias iterativas

Interrupção do laço com uso de *flag*

Algoritmos: Estruturas de Repetição

Planejando laços para
solucionar problemas

Mais exemplos

Aplicação em séries

Escrever os 10 primeiros termos da série:

2, 5, 8, ...

Aplicação em séries

Escrever os 10 primeiros termos da série:

1, 1, 1, 1, ...

1 2 4 8

Aplicação em séries

Calcular o valor de H, com precisão de 10 termos, onde:

$$H = \frac{1}{1} + \frac{2}{9} + \frac{3}{8} + \frac{4}{7}, \dots$$

Aplicação em séries

Escrever os 10 primeiros termos Fibonacci