INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Pelotas

EDUCAÇÃO PÚBLICA 100% GRATUITA

Estrutura de Dados

Aula 1

Funções

Passagem de Parâmetros por Valor

Revisão

Curso Superior de Tecnologia em Sistemas para Internet

REVISÃO DE LÓGICA E PROGRAMAÇÃO

(Baseada nos erros mais comuns na resolução dos exercícios)

1. Revisão de Funções

PROBLEMA 1:

- ✓ Escreva uma função que imprima uma linha com 22 asteriscos.
- ✓ A linha deve ser impressa com repetição.
- ✓ Imprimir a tela abaixo usando a função.

```
**********************
IFSUL
**********************
CSTSI

Logica de programação
**********************
Aluno: Fulano
**********************
```

1. Funções

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

void linhaDeAsteriscos(void);

int main()
{
    linhaDeAsteriscos();
    printf("\nIFSUL\n");
    linhaDeAsteriscos();
    printf("\nCSTSI\n\n");
    printf("Lógica de programação\n");
    linhaDeAsteriscos();
    printf("\nAluno: Fulano\n");
    linhaDeAsteriscos();
    system("pause");
    return 0;
}

void linhaDeAsteriscos()
{
    int i;

    for (i=1; i<=22; i++)
        printf("*");
}
```

Protótipo da função

Programa principal (função principal)

Chamada da função

Nome da função

Cabeçalho da função

Variável local

Definição da função (corpo)

```
**********************
IFSUL
**********************
CSTSI

Logica de programação
**********************
Aluno: Fulano
**********************
```

2. Revisão de Variáveis Locais

PROBLEMA 2:

- Escreva um programa para ler um inteiro Q e imprimir Q linhas de 22 asteriscos usando a função linhaDeAsteriscos() desenvolvida anteriormente.

```
Informe Q: 4

**********************
**********************
**********************
**********************
```

2. Variáveis Locais:

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

void linhaDeAsteriscos(void);

int main()
{
    int i,q;  ————————> Variáveis locais

    printf("Informe Q:");
    scanf("%d",&q);
    for (i=1; i<=q; i++) {
       linhaDeAsteriscos();
       printf("\n");
    }
    system("pause");
    return 0;
}

void linhaDeAsteriscos()
{
    int i;    ————————> Variável local

    for (i=1; i<=22; i++)
       printf("*");
}
```

- São visíveis apenas no local onde são declaradas.
- São criadas quando a execução da função inicia e destruídas quando termina.

```
Informe Q: 6

**********************
**********************
**********************
**********************
**********************
**********************
```

2. Variáveis Locais

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

void exibeNumero(void);

int main()
{
    int num;

    num = 30;
    exibeNumero();
    printf("Número (main):%d\n",num);
    system("pause");
    return 0;
}

void exibeNumero()
{
    num = 40;
    printf("Número (função):%d\n",num);
}
```

Variável Local não declarada!

- Estão isoladas dentro da função onde foram declaradas.

Esse programa não compila.

Por quê???

2. Variáveis Locais

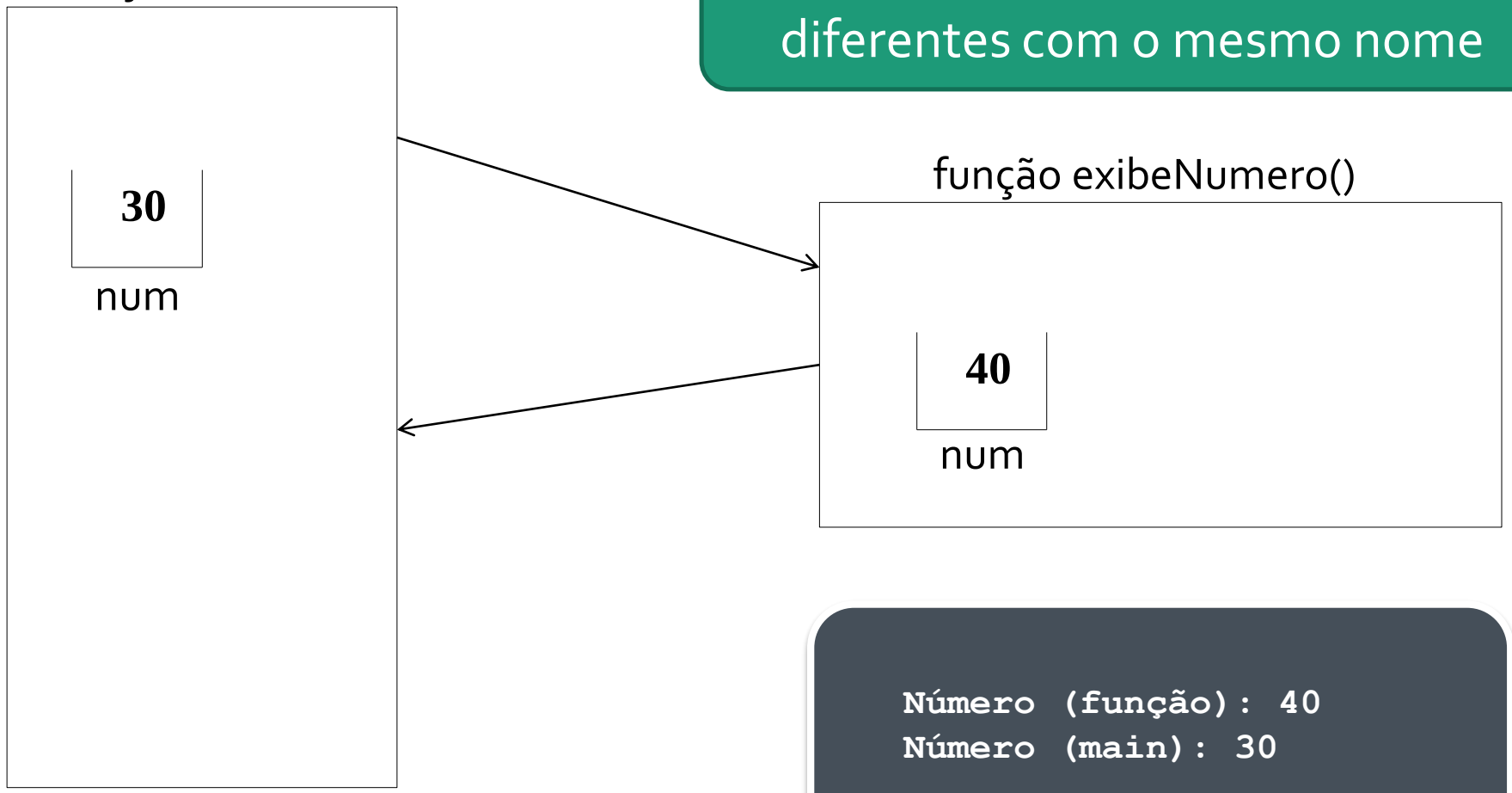
```c
#include <stdio.h>
#include <stdlib.h>

void exibeNumero(void);

int main()
{
    int num;

    num = 30;
    exibeNumero();
    printf("Número (main):%d\n",num);
    system("pause");
    return 0;
}

void exibeNumero()
{
    int num;

    num = 40;
    printf("Número (função):%d\n",num);
}
```

?

```
Número (função): 40
Número (main): 30
```

2. Variáveis Locais

São duas variáveis "num" diferentes com o mesmo nome




```
Número (função): 40
Número (main): 30
```

PROBLEMA 3:

Escreva um programa para exibir a seguinte tela. Cada linha de asteriscos deve ser impressa com uma chamada à função **linhaDeAsteriscos()**.

```
*****
IFSUL
************************
```

- Como permitir que a função **linhaDeAsteriscos()** exiba uma quantidade qualquer de asteriscos?

Funções

Tentativa 1

Esse programa não compila.

Por quê???

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

void linhaDeAsteriscos(void);

int main()
{
        int n;

        n=5;
        linhaDeAsteriscos();
        printf("\nIFSUL\n");
        n=22;
        linhaDeAsteriscos();
        system("pause");
        return 0;
}

void linhaDeAsteriscos()
{
        int i;

        for (i=1; i<=n; i++)
            printf("*");
}
```

Variável Local não declarada!

Funções

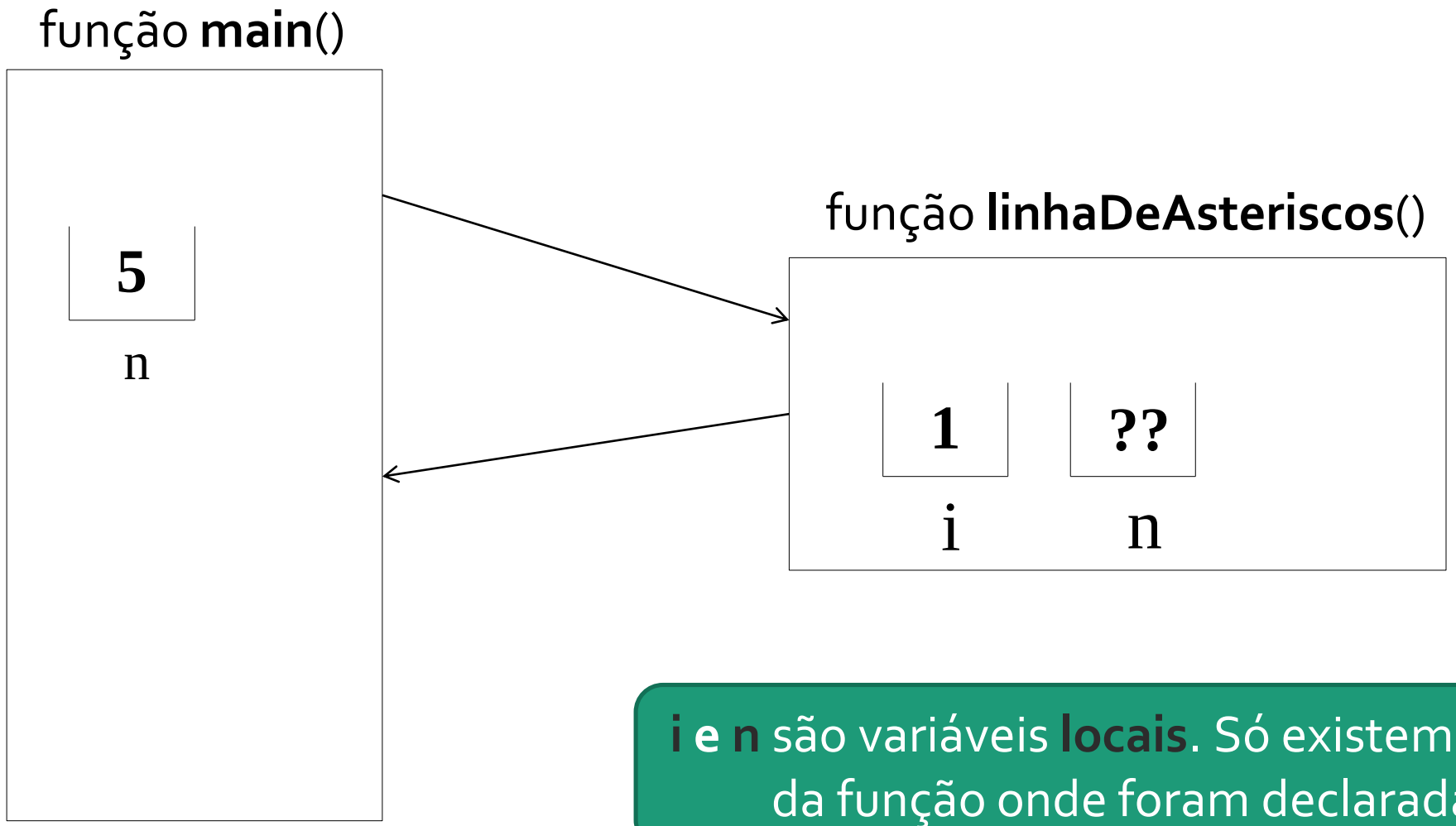
Tentativa 2

Esse programa compila, mas não funciona!

Por quê???

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

void linhaDeAsteriscos(void);

int main()
{
int n;

n=5;
linhaDeAsteriscos();
printf("\nIFSUL\n");
n=22;
linhaDeAsteriscos();
system("pause");
return 0;
}

void linhaDeAsteriscos()
{
int i,n;

for (i=1; i<=n; i++)
  printf("*");
}
```

Variável sem atribuição de valor!



i e n são variáveis **locais**. Só existem dentro da função onde foram declaradas.

Passagem de parâmetros

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

void linhaDeAsteriscos(int n);

int main()
{
   linhaDeAsteriscos(5);
   printf("\nIFSUL\n");
   linhaDeAsteriscos(22);
   system("pause");
   return 0;
}

void linhaDeAsteriscos(int n)
{
int i;

for (i=1; i<=n; i++)
  printf("*");
}
```

Argumento

Parâmetro

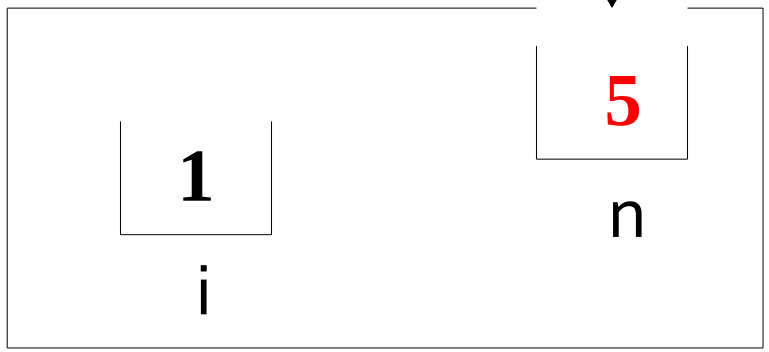
Como quebrar o isolamento das variáveis?

Declarando **parâmetros de entrada.**

3. Passagem de parâmetros

```
int main()
{
linhaDeAsteriscos(5);
...
```

```
linhaDeAsteriscos(int n)
```



O argumento 5 é passado para o parâmetro n declarado na função **linhaDeAsteriscos**.

A variável n continua existindo apenas na função onde ela foi declarada.

3. Passagem de parâmetros

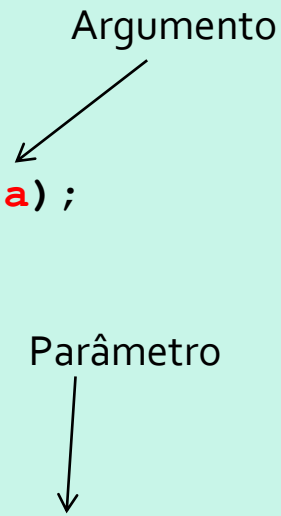
PROBLEMA:

Escreva um programa para exibir a seguinte tela. Cada linha de asteriscos deve ser impressa com uma chamada à função **linhaDeAsteriscos**().

```
*
**
***
****
*****
```

3. Passagem de parâmetros

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

void linhaDeAsteriscos(int n);

int main()
{
    int a;

    for (a=1; a<=5; a++)
    {
        linhaDeAsteriscos(a);
        printf("\n");
    }
    system("pause");
    return 0;
}

void linhaDeAsteriscos(int n)
{
    int i;

    for (i=1; i<=n; i++)
        printf("*");
}
```



- Uma variável pode ser utilizada como argumento. O **valor** da variável **a** é **copiado** para a variável **n**.
- A variável **a** continua existindo apenas na função onde ela foi declarada.
- Uma **cópia** da variável **a** é passada para a variável **n**.

3. Passagem de parâmetros

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

void alteraNumero(int num);

int main()
{
    int num;

    num = 30;
    printf("Número (main)antes:%d\n",num);
    alteraNumero(num);
    printf("Número (main) depois:%d\n",num);
    system("pause");
    return 0;
}

void alteraNumero(int num)
{
    num = 40;
    printf("Número (função):%d\n",num);
}
```

?

```
Número (main) antes: 30
Número (função): 40
Número (main) depois: 30
```

4. Funções com retorno de valor

- Como permitir que uma função **devolva** o valor de uma variável para o local de onde ele foi chamado?

PROBLEMA:

a) Escreva uma função chamada **calculaSerie** que receba como entrada um inteiro que representa a quantidade de termos e **retorne** o valor de S para a série abaixo.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots$$

b) Escreva um programa que calcule e imprima o valor de **S** para 4 termos da série acima. O valor deve ser obtido com a chamada à função **calculaSerie**.

4. Funções com retorno de valor

O valor retornado é armazenado na variável

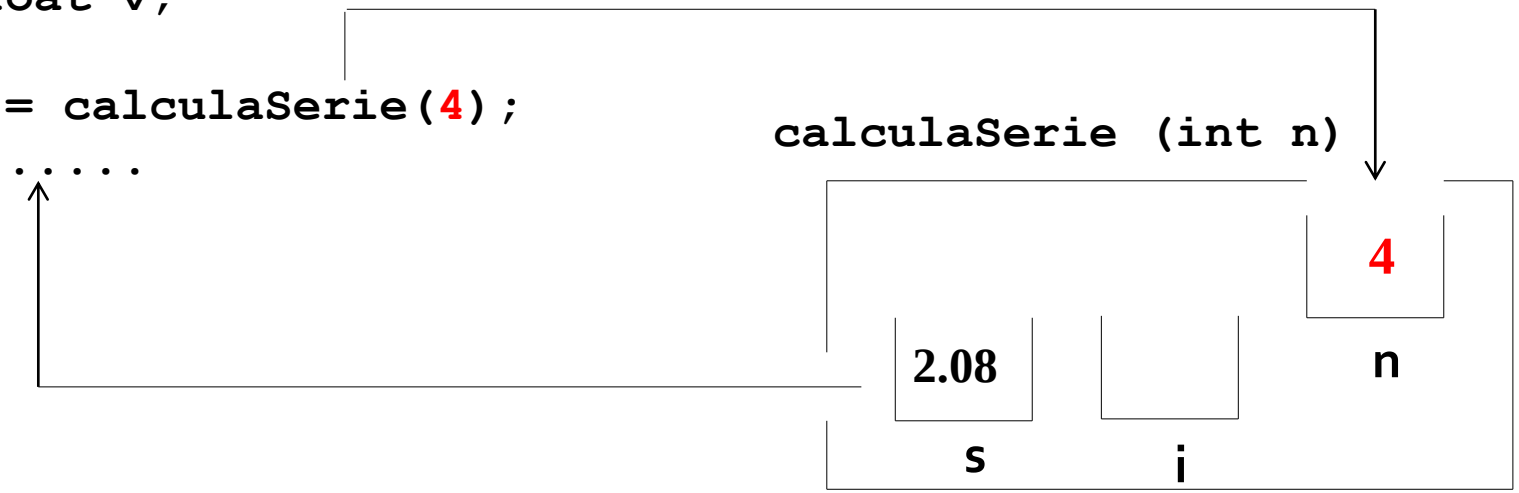
Tipo retornado pela função

Comando que retorna o valor

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

float calculaSerie(int n);

int main()
{
    float v;

    v = calculaSerie(4);
    printf("Valor: %f\n",v);
    system("pause");
    return 0;
}

float calculaSerie(int n)
{
    int a;
    float s;

    s=0;
    for (a=1; a<=n; a++)
        s = s + (float) 1/a;
    return s;
}
```

4. Funções com retorno de valor

```
main()
{
float v;

v = calculaSerie(4);
........
```



O argumento 4 é passado para o parâmetro n declarado na função **calculaSerie**.

O valor de s é retornado para o ponto onde a função **calculaSerie** foi chamada.

4. Funções com retorno de valor

```
#include <stdio.h>

void main()
{
    printf("Alô mundo");
}
```

Usado para indicar que a função main() **não** retorna valor

```
#include <stdio.h>

int main()
{
    printf("Alô mundo");
    return 0;
}
```

Indica que a função main() encerrou corretamente.

Como resolver o problema a seguir?

PROBLEMA:

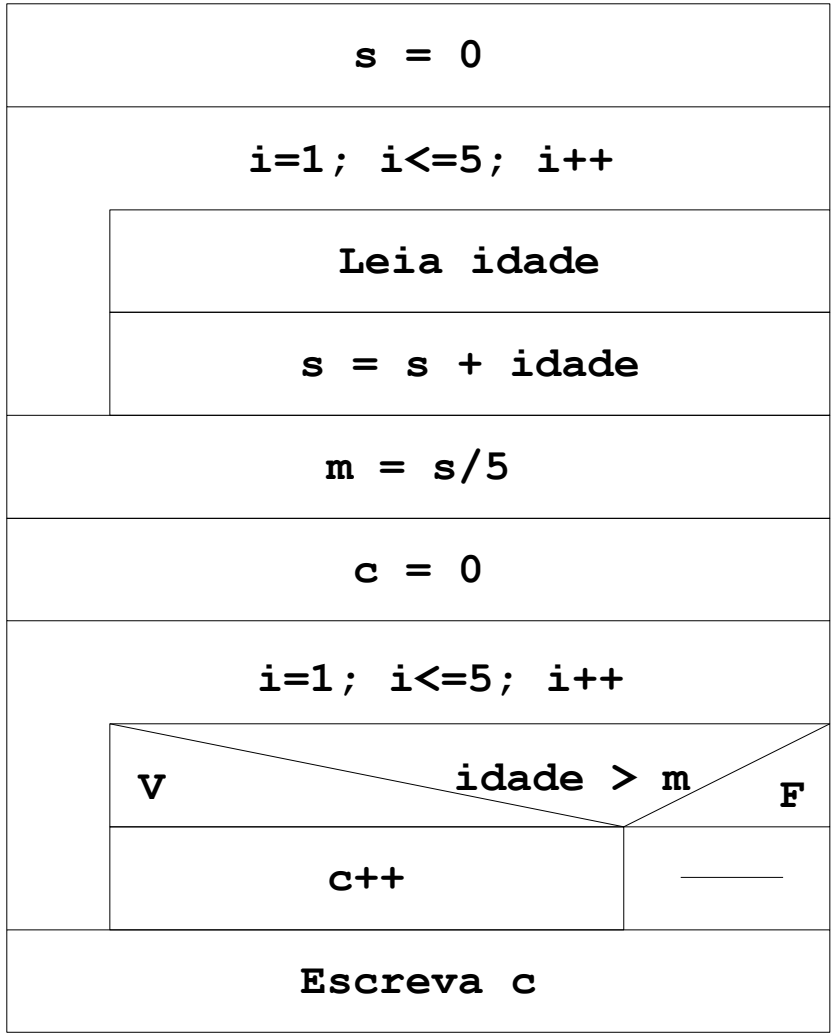
- Escreva um algoritmo para ler a idade de cada componente de um grupo de 5 alunos. Contar e escrever quantos alunos possuem idade acima da média de idade do grupo.

[Entrada]	[Saída]
21	
10	
9	
17	
18	3

- OBS: A média de idade do grupo é 15

Solução incorreta 1

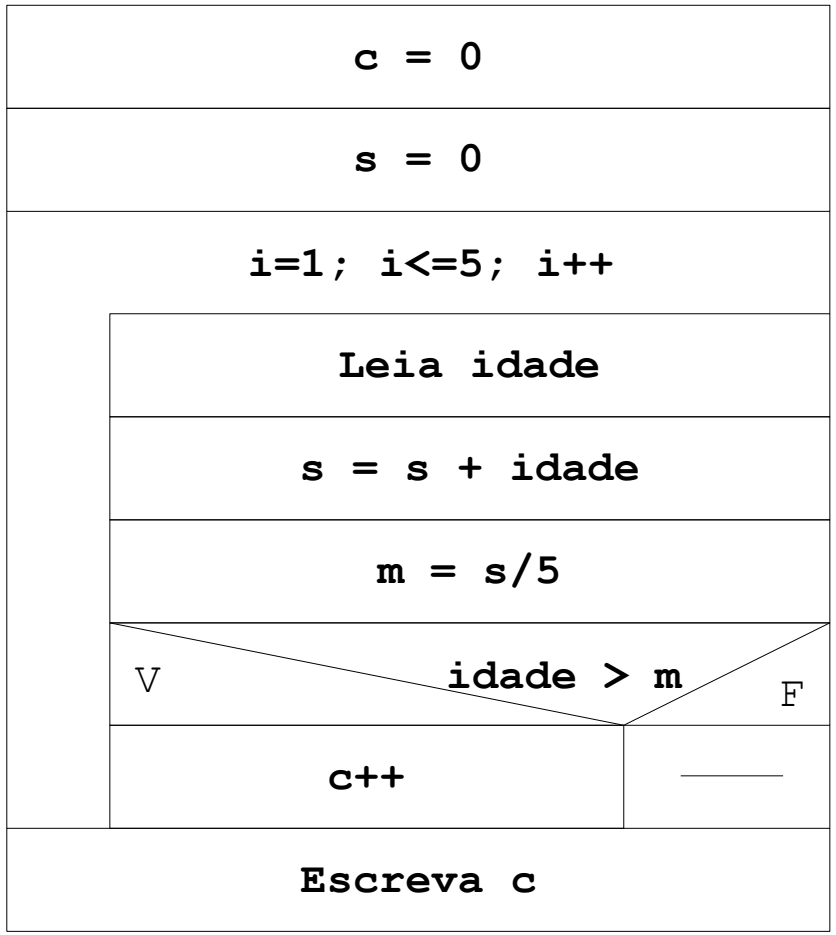
- Por que está incorreta?



Variável **idade** com atribuição apenas da última entrada

Solução incorreta 2

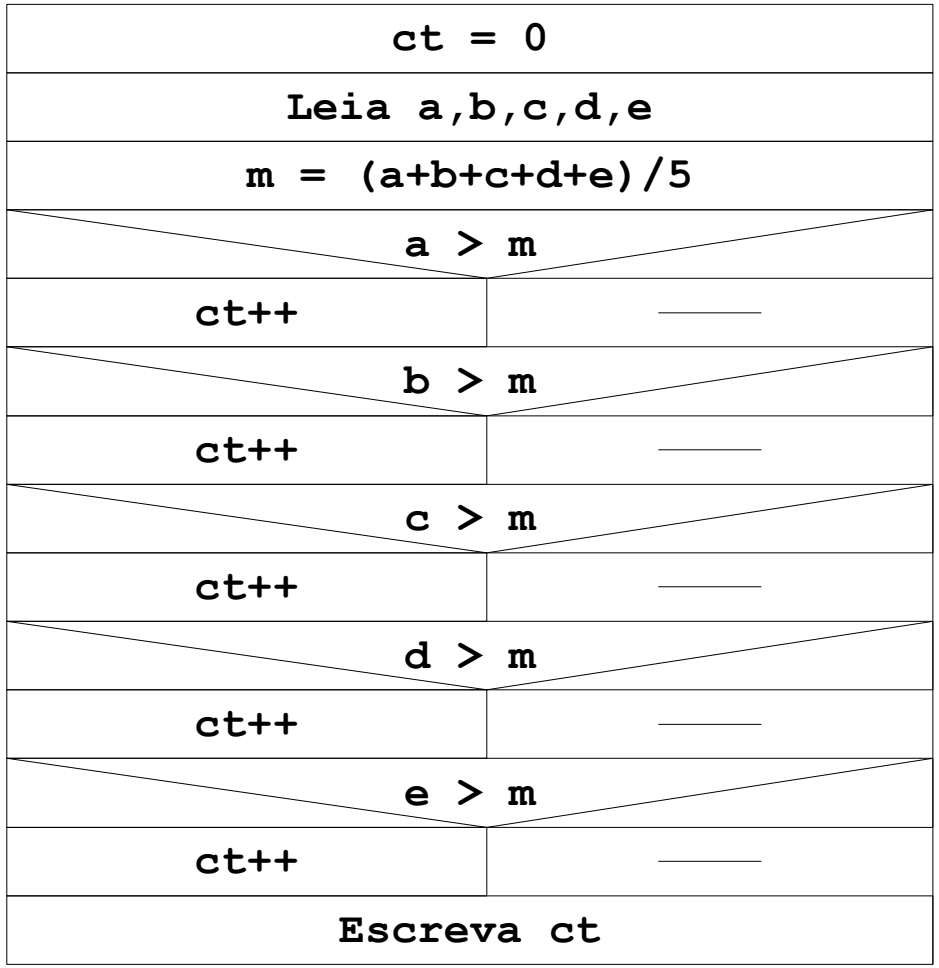
- Por que está incorreta?



Variável **m** e **s** com atribuições incompletas

Solução 3

- Correta, mas inadequada

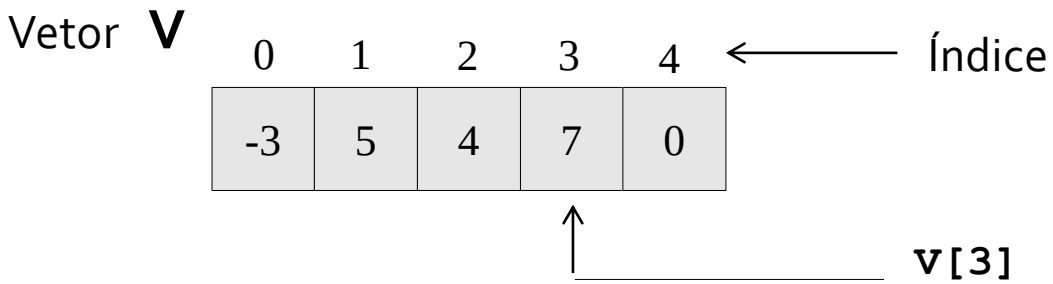


5. Variáveis indexadas (Vetores)

- É um conjunto de variáveis do **mesmo tipo** que compartilham um **mesmo nome**. Índices são associados a este nome com a finalidade de permitir a individualização dos elementos do conjunto.

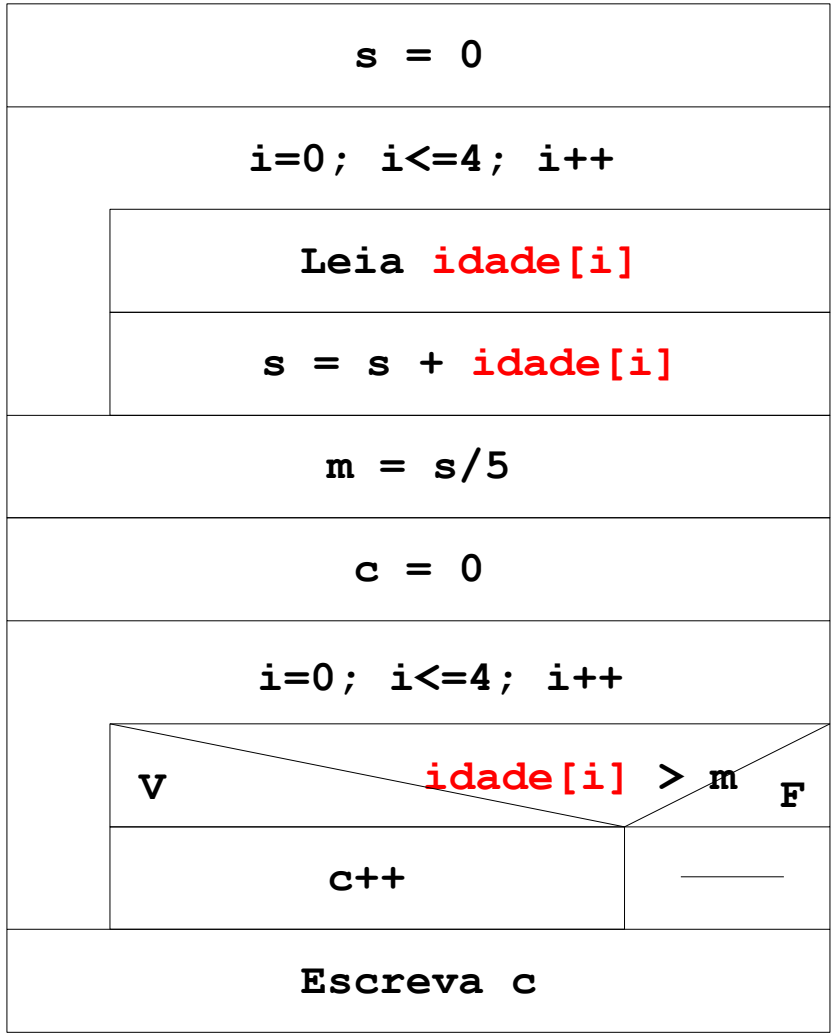
- Quando possui apenas um índice (uma dimensão) chamamos de matriz unidimensional ou **vetor**.

- Ex: Um vetor V de 5 elementos



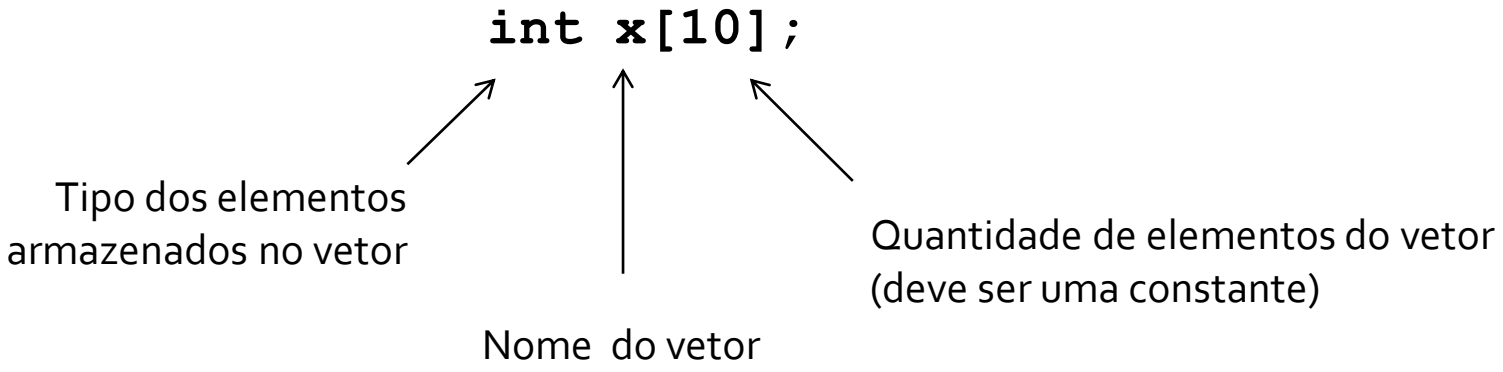
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int idade[5],i,s,c;
    float m;
    s = 0;
    for (i=0; i<=4; i++) {
        printf("Informe a idade %d:",i);
        scanf("%d",&idade[i]);
        s = s + idade[i];
    }
    m = (float) s/5;
    c = 0;
    for (i=0; i<=4; i++)
        if (idade[i]>m)
            c++;
    printf("%d\n",c);
    system("pause");
    return 0;
}
```

Resolvendo o problema proposto

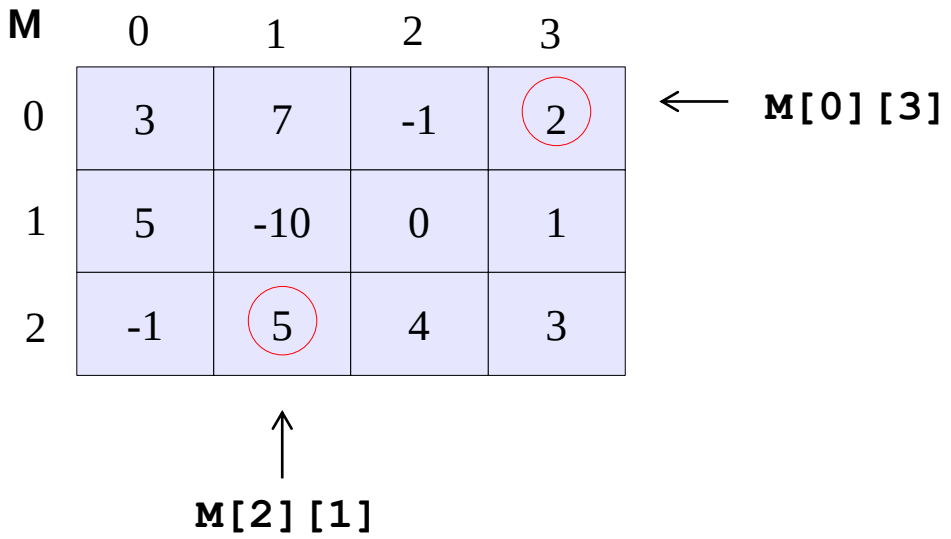


Observações sobre variáveis indexadas - vetores

- Ao acessar os elementos de um vetor não utilizar índices fora da faixa.
- O índice deve ser um inteiro (constante, variável ou expressão)
- O primeiro elemento do vetor possui índice **zero**
- Um vetor pode ser declarado para armazenar valores de qualquer tipo. Ex: **float vet[100];**
- Todos os elementos do vetor são do mesmo tipo.
- A constante definida na declaração indica a quantidade máxima de elementos de um vetor.



6. Matrizes



Declaração:

```
int M[3][4]
```

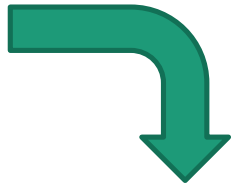
Qde. linhas

Qde. colunas

6. Matrizes

Considerando que a matriz M possui os valores indicados abaixo:

	0	1	2	3
0	1	5	9	-1
1	2	6	10	14
2	3	7	11	15
3	4	8	20	32



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	20
3	-1	14	15	32

Qual o conteúdo da matriz após a execução do seguinte trecho de programa?

```
...
for (i=0; i<=2; i++)
  for (j=i+1; j<=3; j++) {
    aux = M[i][j];
    M[i][j] = M[j][i];
    M[j][i] = aux;
  }
...
```

INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Pelotas

EDUCAÇÃO PÚBLICA 100% GRATUITA

Estrutura de Dados

Aula 1

Funções

Passagem de Parâmetros por Valor

Revisão

Curso Superior de Tecnologia em Sistemas para Internet