

INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Estrutura de Dados

Aula 1

Funções

Passagem de Parâmetros por Valor
Revisão



REVISÃO DE LÓGICA E PROGRAMAÇÃO

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  
*****
```

Funções

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

```
void linhaDeAsteriscos(void);
```

 Protótipo da função

```
int main()
```

 Programa principal (função principal)

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

 Chamada da função
Nome da função

```
void linhaDeAsteriscos()
```

 Cabeçalho da função

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

 Variável local
Definição da função (corpo)

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

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

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

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



```
#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("*");
}
```

Variáveis Locais:

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

Informe Q: 6

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



Variáveis Locais

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

void exhibeNumero(void) ;

int main()
{
    int num;

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

void exhibeNumero()
{
    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???

```
#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);
}
```

Variáveis Locais

?

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



EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Variáveis Locais

função main()

30

num

São duas variáveis “num”
diferentes com o mesmo nome

função exibeNumero()

40

num

```
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!



EDUCAÇÃO
PÚBLICA
100%
GRATUITA

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!



EDUCAÇÃO
PÚBLICA
100%
GRATUITA

função **main()**

5

n

função **linhaDeAsteriscos()**

1

??

i

n

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.



Passagem de parâmetros

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

```
linhaDeAsteriscos(int n)
```

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

	5
1	n
i	

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



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()**.

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


Passagem de parâmetros

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

void linhaDeAsteriscos(int n);

int main()
{
    int a;
    // Argumento
    for (a=1; a<=5; a++)
    {
        linhaDeAsteriscos(a);
        printf("\n");
    }
    system("pause"); // Parâmetro
    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**.



Passagem de parâmetros

```
#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
```



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.

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

- 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**.



- # 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;
}
```



Funções com retorno de valor

```
main()  
{  
  float v;  
  
  v = calculaSerie(4);  
  .....  
}
```

`calculaSerie (int n)`

`4`

`2.08` `n`

`s` `i`

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.



- ## 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?

```
s = 0
```

```
i=1; i<=5; i++
```

```
Leia idade
```

```
s = s + idade
```

```
m = s/5
```

```
c = 0
```

```
i=1; i<=5; i++
```

```
v          idade > m          F
```

```
c++
```

```
Escreva c
```

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



Solução incorreta 2

- Por que está incorreta?

```
c = 0
```

```
s = 0
```

```
i=1; i<=5; i++
```

```
Leia idade
```

```
s = s + idade
```

```
m = s/5
```

```
V          idade > m          F
```

```
c++
```

```
Escreva c
```

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



Solução 3

- Correta, mas inadequada

```
ct = 0
Leia a,b,c,d,e
m = (a+b+c+d+e)/5
a > m
ct++
b > m
ct++
c > m
ct++
d > m
ct++
e > m
ct++
Escreva ct
```



Variáveis indexadas

- 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 possuí apenas um índice (uma dimensão) chamamos de matriz unidimensional ou **vetor**.
- Ex: Um vetor V de 5 elementos

Vetor V	0	1	2	3	4	Índice
	-3	5	4	7	0	
						v[3]

```

#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

```

s = 0

i=0; i<=4; i++

Leia idade[i]

s = s + idade[i]

m = s/5

c = 0

i=0; i<=4; i++

V          idade[i] > m          F

c++

Escreva c

```

Observações sobre variáveis indexadas

- 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.

```
int x[10] ;
```

Tipo dos elementos
armazenados no vetor

Nome do vetor

Quantidade de elementos do vetor
(deve ser uma constante)

Matrizes

M	0	1	2	3
0	3	7	-1	2
1	5	-10	0	1
2	-1	5	4	3

`M[0][3]`

`M[2][1]`

Declaração:

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

Qde. linhas

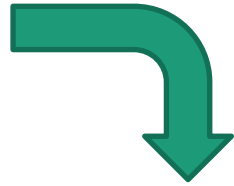
Qde. colunas



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

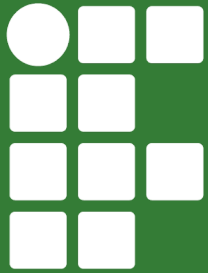


	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	20
3	-1	14	15	32

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