Programação 2

Passagem de parâmetros

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`**.

```
***
TRO
*******
```

Como permitir que a função **`linhaDeAsteriscos`** exiba uma quantidade qualquer de asteriscos?

Passagem de parâmetros

Tentativa 1

Funciona???

```
#include <stdio.h>

void linhaDeAsteriscos(void);

main()
{
int n;
n=3;
linhaDeAsteriscos();
printf("\nTRO\n");
n=7;
linhaDeAsteriscos();
}

void linhaDeAsteriscos(void)
{
int i;

                    n
for (i=1; i<= 10; i++)
  printf("*");
}
```

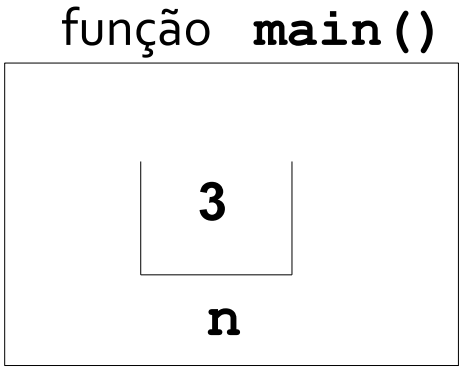
Passagem de parâmetros

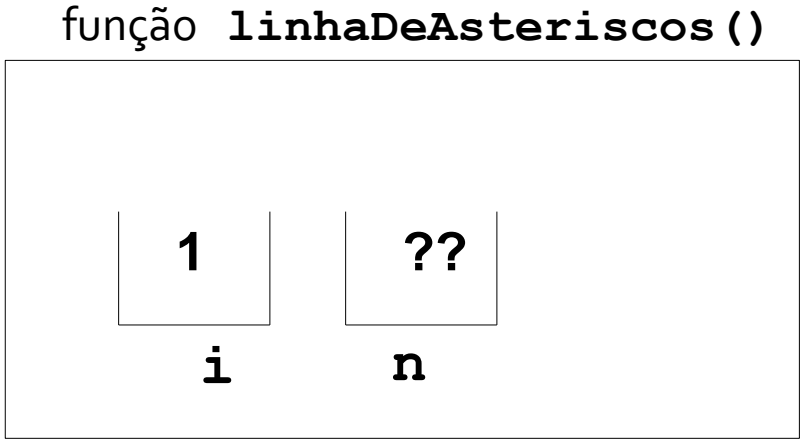
Tentativa 2

E agora???

```
#include <stdio.h>

void linhaDeAsteriscos(void);

main()
{
int n;
n=3;
linhaDeAsteriscos();
printf("\nTRO\n");
n=7;
linhaDeAsteriscos();
}

void linhaDeAsteriscos(void)
{
int i,n;

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

Passagem de parâmetros






```
...
main()
{
int n;
n=3;
linhaDeAsteriscos();
printf("\nTRO\n");
...
}
```

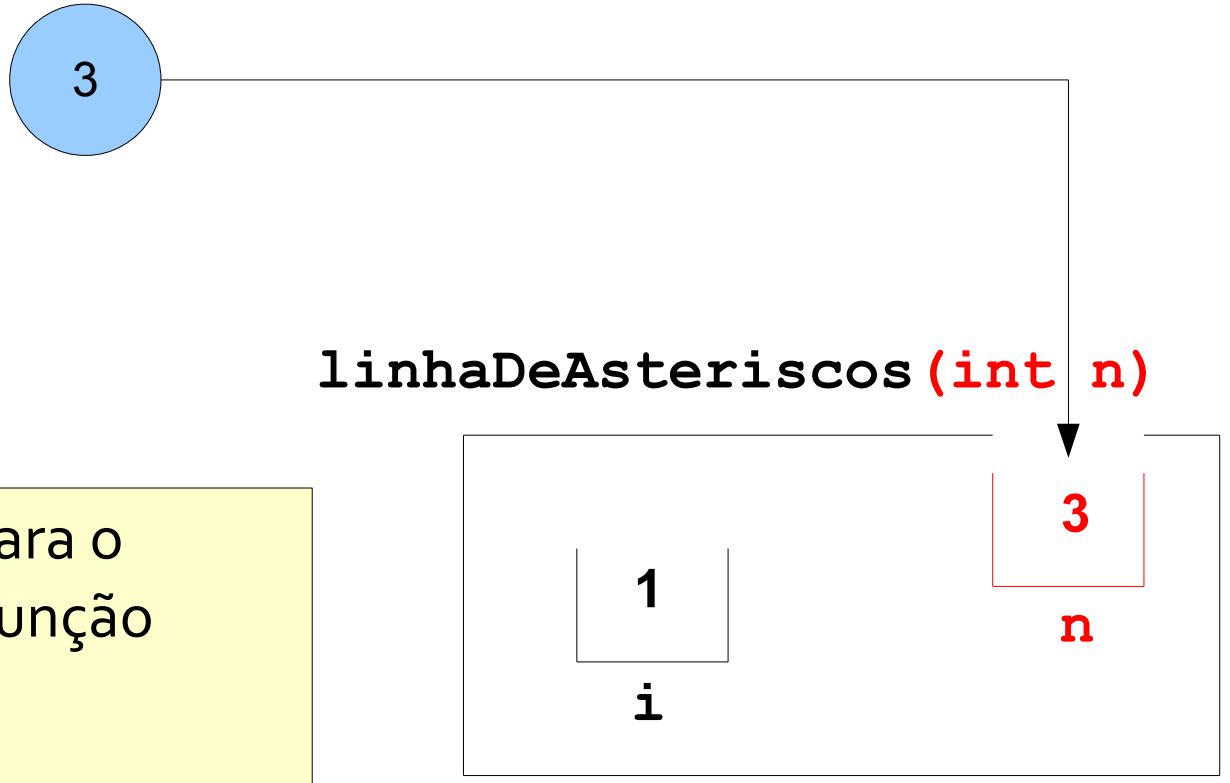
i e **n** são variáveis **locais**. Só existem dentro da função onde foram declaradas.

Como quebrar o **isolamento** das variáveis?

Passagem de parâmetros

Com passagem de parâmetros.

```
...
main()
{
linhaDeAsteriscos(3);
printf("\nTRO\n");
...
}
```

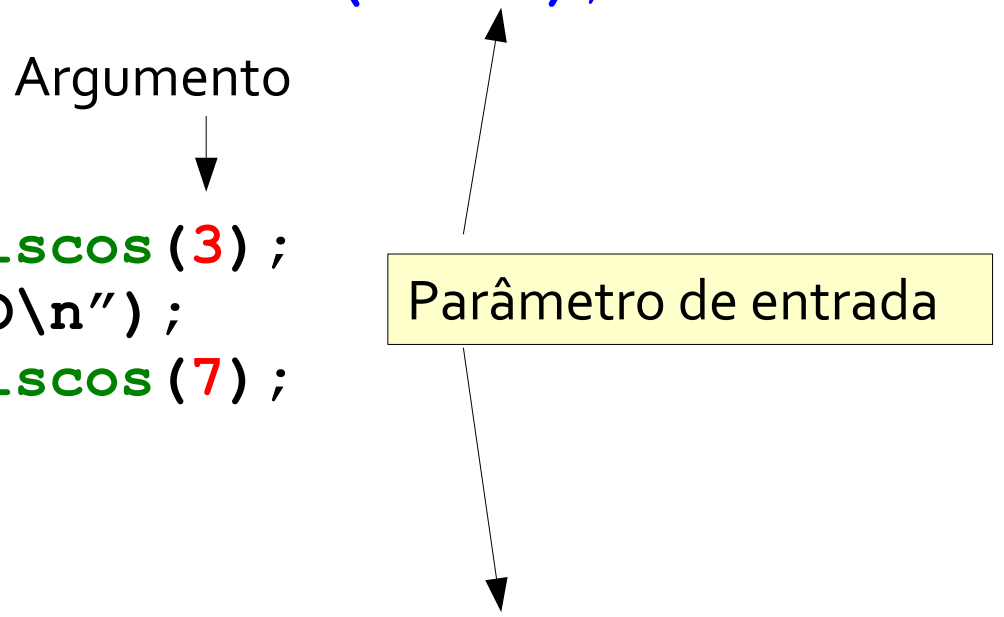


O argumento **3** é passado para o parâmetro **n** declarado na função **linhaDeAsteriscos**.

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

Passagem de parâmetros

```
#include <stdio.h>


void linhaDeAsteriscos(int n);

main()
{
linhaDeAsteriscos(3);
printf("\nTRO\n");
linhaDeAsteriscos(7);
}


void linhaDeAsteriscos(int n)
{
int i;

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



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

Primeira solução

```
#include <stdio.h>

void linhaDeAsteriscos(int n);

main()
{
linhaDeAsteriscos(1); printf("\n");
linhaDeAsteriscos(2); printf("\n");
linhaDeAsteriscos(3); printf("\n");
linhaDeAsteriscos(4); printf("\n");
}

void linhaDeAsteriscos(int n)
{
int i;

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

Sugestão para reduzir o código??

Passagem de parâmetros

Segunda solução

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.

```
#include <stdio.h>

void linhaDeAsteriscos(int n);

main()
{
int a;

for (a=1; a<=4; a++)
   {
   linhaDeAsteriscos(a);
   printf("\n");
   }
}

void linhaDeAsteriscos(int n)
{
int i;

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

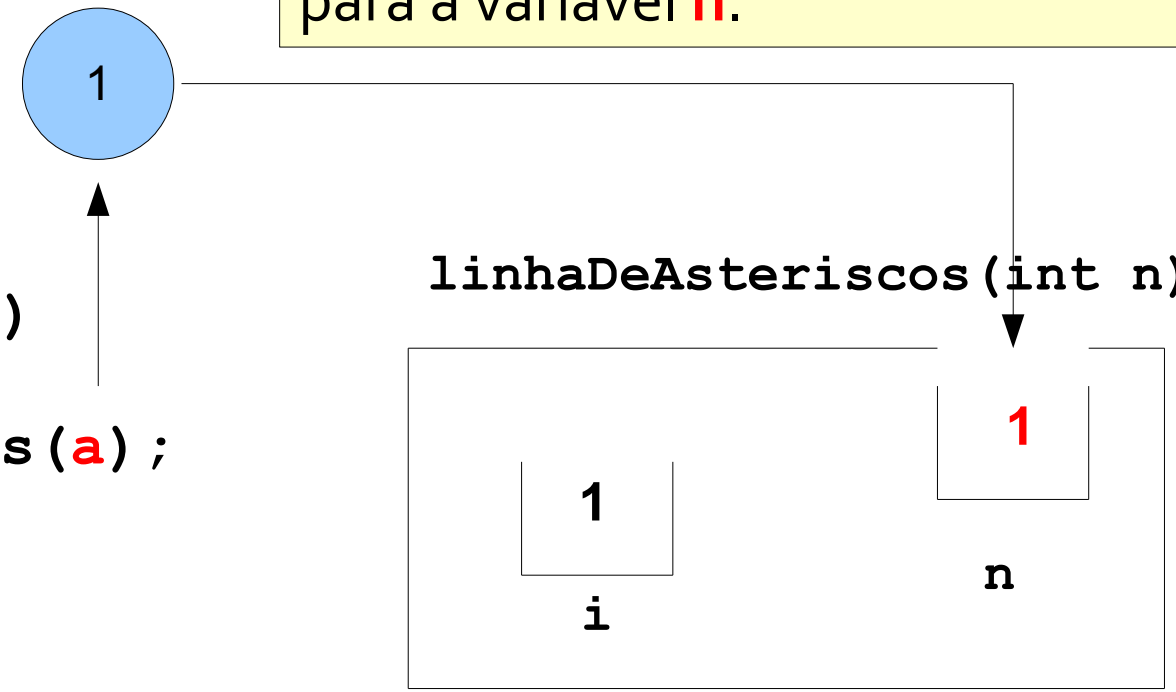
Argumento

Parâmetro

Passagem de parâmetros

O **conteúdo** da variável **a** é passado para a variável **n**.

```
main()
{
int a;

for (a=1; a<=5; a++)
    {
    linhaDeAsteriscos(a);
    printf("\n");
    }
...
```

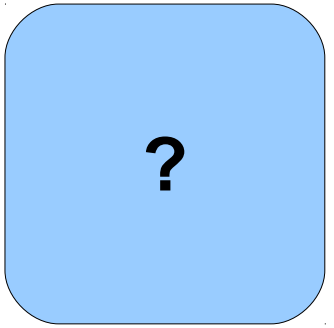


Uma expressão também pode ser usada como argumento.

```
...
linhaDeAsteriscos(a+4);
...
```

Passagem de parâmetros

```
#include <stdio.h>

void alteraNumero(int n);

main()
{
int num;

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

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



Passagem de parâmetros

```
#include <stdio.h>

void alteraNumero(int num);

main()
{
int num;

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

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

?

Funções com mais de um parâmetro

```
#include <stdio.h>


void linhaDeCaracteres(int n,char ch);

main()
{
linhaDeCaracteres(3,'=');
printf("\nTRO\n");
linhaDeCaracteres(6,'-');
}

void linhaDeCaracteres(int n,char ch)
{
int i;

for (i=1; i<=n; i++)
  printf("%c",ch);
}
```

```
===
TRO
------
```

Argumentos

Parâmetros

A **ordem** dos argumentos determina a relação entre os argumentos e parâmetros.

Funções com mais de um parâmetro

indica que a função **não** possui parâmetros

Os tipos devem ser declarados de forma independente mesmo para parâmetros de mesmo tipo

```
#include <stdio.h>

void linhaDezAsteriscos(void);
void exibeSoma(float a,float b);

main()
{
float x,y;

printf("Informe um valor:");
scanf("%f",&x);
printf("Informe outro:");
scanf("%f",&y);
linhaDezAsteriscos();
printf("\n");
exibeSoma(x,y);
linhaDezAsteriscos();
}
```

```
void linhaDezAsteriscos(void)
{
int i;

for (i=1; i<=10; i++)
  printf("*");
}

void exibeSoma(float a,float b)
{
float soma;

soma = a+b;
printf("Soma: %f\n",soma);
}
```