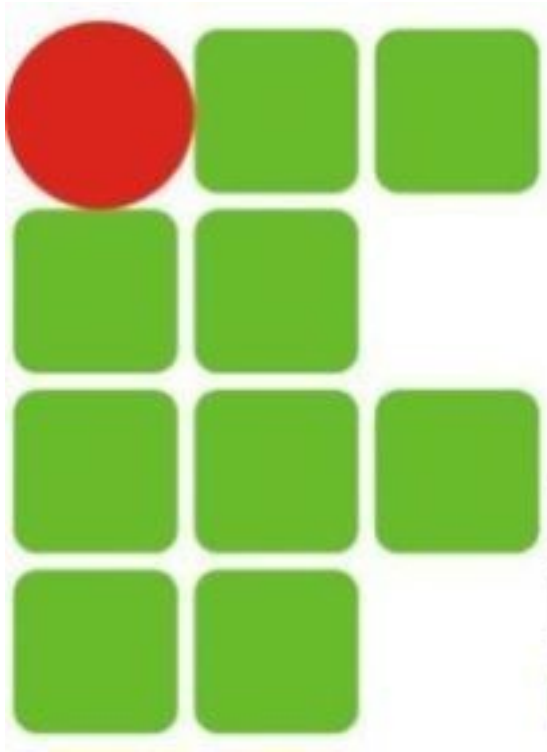


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA

LINGUAGEM DE PROGRAMAÇÃO

FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Uma função nada mais é do que um bloco de código (ou seja, declarações e outros comandos) que pode ser nomeado e chamado de dentro de um programa
- uma função é uma sequência de comandos que recebe um nome e pode ser chamada de qualquer parte do programa, quantas vezes forem necessárias, durante a sua execução.
- programador não precisa saber o código contido dentro das funções de entrada e saída para utilizá-las. Basta saber seu nome e como utilizá-la.
- Exemplos: scanf(), printf()
- Duas são as principais razões para o uso de funções:
  - Estruturação dos programas (programas contruídos a partir de pequenos blocos).
  - Reutilização de código (o trecho de código pode ser utilizado diversas vezes).

FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- a declaração de uma função pelo programador segue esta forma geral:

```
tipo_retornado nome_função (lista_de_parâmetros){
   sequência de declarações e comandos
}
```

- O nome_função é como aquele trecho de código será conhecido dentro do programa.
- Para definir esse nome, valem basicamente as mesmas regras para se definir uma variável.

- Com relação ao local de declaração de uma função, ela deve ser definida ou declarada antes de ser utilizada, ou seja, antes da cláusula main()

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

int Square (int a){
  return (a*a);
}

int main(){
  int n1,n2;
  printf("Entre com um numero: ");
  scanf("%d", &n1);
  n2 = Square(n1);
  printf("O seu quadrado vale: %d\n", n2);
  system("pause");
  return 0;
}
```

FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- a declaração de uma função pelo programador segue esta forma geral:

```
tipo_retornado nome_função (lista_de_parâmetros){
   sequência de declarações e comandos
}
```

- O nome_função é como aquele trecho de código será conhecido dentro do programa.
- Para definir esse nome, valem basicamente as mesmas regras para se definir uma variável.

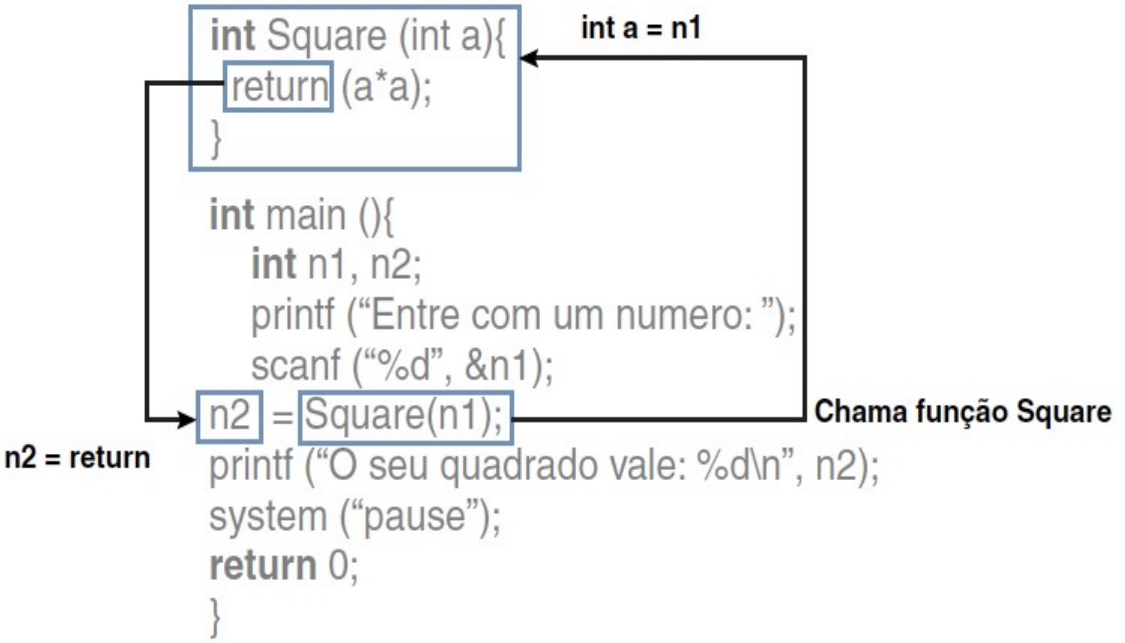
- Com relação ao local de declaração de uma função, ela deve ser definida ou declarada antes de ser utilizada, ou seja, antes da cláusula main()

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

int Square (int a){
  return (a*a);
}

int main(){
  int n1,n2;
  printf("Entre com um numero: ");
  scanf("%d", &n1);
  n2 = Square(n1);
  printf("O seu quadrado vale: %d\n", n2);
  system("pause");
  return 0;
}
```

FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- O funcionamento de uma função:
  - O código do programa é executado até encontrar uma chamada de função.
  - O programa é então interrompido temporariamente, e o fluxo do programa passa para a função chamada
  - Se houver parâmetros na função, os valores da chamada da função são copiados para os parâmetros no código da função.
  - Os comandos da função são executados.
  - Quando a função termina, o programa volta ao ponto em que foi interrompido para continuar sua execução normal.
  - •• Se houver um comando return, o valor dele será copiado para a variável que foi escolhida para receber o retorno da função.



FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Parâmetros
  - são o que o programador utiliza para passar a informação de um trecho de código para dentro da função.
  - são uma lista de variáveis, separadas por vírgula, em que são especificados o tipo e o nome de cada variável passada para a função.

```
tipo_retornado nome_função (tipo nome1, tipo nome2, ... , tipo nomeN){
   sequência de declarações e comandos
}
```

```
//Declaração CORRETA de parâmetros
int soma(int x, int y){
      return x + y;
}

//Declaração ERRADA de parâmetros
int soma(int x, y){
      return x + y;
}
```

FUNÇÕES EM C

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Dependendo da função, ela pode não possuir nenhum parâmetro. Nesse caso, pode-se optar por duas soluções:
  - Deixar a lista de parâmetros vazia: void imprime().
  - Colocar void entre parênteses: void imprime(void).

EXERCÍCIO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Altere o programa abaixo criando uma função para calcular o fatorial do número

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

int main(){
  printf("Digite um numero inteiro positivo:");
  int x;
  scanf("%d",&x);
  int i,f = 1;
  for (i=1; i<=x; i++)
      f = f * i;

  printf("O fatorial de %d eh: %d\",x,f);
  system("pause");
  return 0;
}
```

EXERCÍCIO - RESPOSTA

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

int main(){
  printf("Digite um numero inteiro positivo:");
  int x;
  scanf("%d",&x);
  int i,f = 1;
  for (i=1; i<=x; i++)
      f = f * i;

  printf("O fatorial de %d eh: %d\",x,f);
  system("pause");
  return 0;
}
```

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

int fatorial (int n){
  int i,f = 1;
  for (i=1; i<=n; i++)
      f = f * i;

  return f;
}

int main(){
  printf("Digite um numero inteiro positivo:");
  int x;
  scanf("%d",&x);
  int fat = fatorial(x);
  printf("O fatorial de %d eh: %d\n",x,fat);

  system("pause");
  return 0;
}
```

FUNÇÕES EM C
RETORNO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- O retorno da função é a maneira como uma função devolve o resultado (se ele existir) da sua execução para quem a chamou
- Uma função pode retornar qualquer tipo válido na linguagem C
- Uma função também pode NÃO retornar um valor. Para isso, basta colocar o tipo void como valor retornado
- O tipo void é conhecido como tipo vazio. Uma função declarada com o tipo void vai apenas executar um conjunto de comando e não devolverá nenhum valor para quem a chamar.

```
#include <stdio.h>
#include <stdlib.h>
void imprime(int n){
  int i;
  for (i=1; i<=n; i++)
      printf("Linha %d \n",i);
}

int main(){
  imprime(5);

  system("pause");
  return 0;
}
```

FUNÇÕES EM C
RETORNO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Se a função não for do tipo void, ela deverá retornar um valor. O comando *return* é utilizado para retornar esse valor para o programa:

```
return expressão;
```

```
#include <stdio.h>
#include <stdlib.h>
int soma(int x, int y){
  return x + y;
}

int main(){
  int a,b,c;
  printf("Digite a: ");
  scanf("%d", &a);
  printf("Digite b: ");
  scanf("%d", &b);
  printf("Soma = %d\n",soma(a,b));
  system("pause");
  return 0;
}
```

FUNÇÕES EM C
RETORNO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Uma função pode ter mais de uma declaração return.

```
int maior(int x, int y){
  if(x > y)
      return x;
  else
      return y;
}
```

- Nesse exemplo, os vários comandos return foram substituídos por uma variável que será retornada no final da função.

```
int maior(int x, int y){
  int z;
  if(x > y)
      z = x;
  else
      z = y;
  return z;
}
```

FUNÇÕES EM C
TIPOS DE PASSAGEM DE PARÂMETROS

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- os parâmetros de uma função são o mecanismo que o programador utiliza para passar a informação de um trecho de código para dentro da função
- existem dois tipos de passagem de parâmetro: passagem **por valor** e **por referência**

FUNÇÕES EM C
PASSAGEM DE PARÂMETROS POR VALOR

- Na linguagem C, os argumentos para uma função são sempre passados por valor, ou seja, uma cópia do dado é feita e passada para a função.
- Esse tipo de passagem de parâmetro é o padrão para todos os tipos básicos predefinidos (int, char, float e double) e estruturas definidas pelo programador (struct).

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

void soma _ mais _ um(int n){
  n = n + 1;
  printf("Dentro da funcao: x = %d\n",n);
}

int main(){
  int x = 5;
  printf("Antes da funcao: x = %d\n",x);
  soma _ mais _ um(x);
  printf("Depois da funcao: x = %d\n",x);
  system("pause");
  return 0;
}
```

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 5
```

FUNÇÕES EM C
PASSAGEM DE PARÂMETROS POR REFERÊNCIA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- existem casos em que é necessário que toda modificação feita nos valores dos parâmetros dentro da função seja repassada para quem a chamou
- Quando se quer que o valor da variável mude dentro da função e essa mudança se reflita fora da função, usam-se passagens de parâmetros por referência.
- Na passagem de parâmetros por referência não se passam para a função os valores das variáveis, mas os endereços das variáveis na memória.
- Para passar um parâmetro por referência, usa-se o operador "*" na frente do nome do parâmetro durante a declaração da função.

Por valor

```
void soma_mais_um(int n){
  n = n + 1;
}
```

Por referência

```
void soma_mais_um(int *n){
  *n = *n + 1;
}
```

Na chamada da função, é necessário utilizar o operador "&" na frente do nome da variável que será passada por referência para a função.

Na passagem de uma variável por referência é necessário usar o operador "*" sempre que se desejar acessar o conteúdo da variável dentro da função.

```
01  include <stdio.h>
02  include <stdlib.h>
03
04  void soma _ mais _ um(int *n){
05     *n = *n + 1;
06      printf("Dentro da funcao: x = %d\n",*n);
07  }
08
09  int main(){
10    int x = 5;
11    printf("Antes da funcao: x = %d\n",x);
12    soma _ mais _ um(&x);
13    printf("Depois da funcao: x = %d\n",x);
14    system("pause");
15    return 0;
16  }
```

Saída

```
Antes da funcao: x = 5
Dentro da funcao: x = 6
Depois da funcao: x = 6
```

FUNÇÕ
PASSAGE

ERAL DE
NCIA E TECNOLOGIA

Por valor	Por referência

`#include <stdio.h>`
`#include <stdlib.h>`
 	
`void Troca(int a,int b){`
`  int temp;`
`  temp = a;`
`  a = b;`
`  b = temp;`
`  printf("Dentro: %d e %d\n",a,b);`
`}`
 	
`int main(){`
`  int x = 2;`
`  int y = 3;`
`  printf("Antes: %d e %d\n",x,y);`
`  Troca(x,y);`
`  printf("Depois: %d e %d\n",x,y);`
`  system("pause");`
`  return 0;`
`}`
	`#include <stdio.h>`
	`#include <stdlib.h>`
	
	`void Troca(int*a,int*b){`
	`  int temp;`
	`  temp = *a;`
	`  *a = *b;`
	`  *b = temp;`
	`  printf("Dentro: %d e %d\n",*a,*b);`
	`}`
	
	`int main(){`
	`  int x = 2;`
	`  int y = 3;`
	`  printf("Antes: %d e %d\n",x,y);`
	`  Troca(&x,&y);`
	`  printf("Depois: %d e %d\n",x,y);`
	`  system("pause");`
	`  return 0;`
	`}`

Saída
`Antes: 2 e 3`
`Dentro: 3 e 2`
`Depois: 2 e 3`
	Saída
	`Antes: 2 e 3`
	`Dentro: 3 e 2`
	`Depois: 3 e 2`