

# FUNÇÕES

Prof. Muriel Mazzetto  
Algoritmos 2

# Problema

2

- Calcular o fatorial de um número.
  - ▣  $2! = 2;$
  - ▣  $3! = 6;$
  - ▣  $4! = 12;$
  - ▣  $5! = 120.$

# Problema

3

## □ Calcular o fatorial de um número.

- $2! = 2;$
- $3! = 6;$
- $4! = 12;$
- $5! = 120.$

```
#include <stdio.h>

int main(void)
{
    int fat, num, i;

    printf("Informe um valor: ");
    scanf("%d", &num);

    fat = 1;
    for(i = num; i > 0 ; i--)
    {
        fat *= i;
    }

    printf("%d! = %d", num, fat);

    return 0;
}
```

# Problema

4

- Calcular o fatorial de dois números e soma-los.
  - $2! + 3! = 8;$
  - $5! + 3! = 126.$

# Problema

5

□ Calcular o fatorial de dois números e soma-los.

□  $2! + 3! = 8;$

□  $5! + 3! = 126.$

```
int main(void)
{
    int fat1, fat2, num1, num2, i;

    printf("Informe um valor: ");
    scanf("%d", &num1);
    fat1 = 1;
    for(i = num1; i > 0 ; i--)
    {
        fat1 *= i;
    }

    printf("Informe um valor: ");
    scanf("%d", &num2);
    fat2 = 1;
    for(i = num2; i > 0 ; i--)
    {
        fat2 *= i;
    }

    printf("%d! + %d! = %d", num1, num2, fat1+fat2);

    return 0;
}
```

# Problema

6

- Calcular o fatorial de quatro números e soma-los.
  - ▣  $2! + 3! + 4! + 5! = 152;$

# Problema

7

- Calcular o fatorial de quatro números e somá-los.
  - ▣  $2! + 3! + 4! + 5! = 152$ ;

```
int main(void)
{
    int fat1, fat2, fat3, fat4, num1, num2, num3, num4;
    int soma, i;

    printf("Informe um valor: ");
    scanf("%d", &num1);
    printf("Informe um valor: ");
    scanf("%d", &num2);
    printf("Informe um valor: ");
    scanf("%d", &num3);
    printf("Informe um valor: ");
    scanf("%d", &num4);

    //FAT DO PRIMEIRO NUMERO
    fat1 = 1;
    for(i = num1; i > 0 ; i--)
        fat1 *= i;
    //FAT DO SEGUNDO NUMERO
    fat2 = 1;
    for(i = num2; i > 0 ; i--)
        fat2 *= i;
    //FAT DO TERCEIRO NUMERO
    fat3 = 1;
    for(i = num3; i > 0 ; i--)
        fat3 *= i;
    //FAT DO QUART NUMERO
    fat4 = 1;
    for(i = num4; i > 0 ; i--)
        fat4 *= i;

    soma = fat1 + fat2 + fat3 + fat4;
    printf("soma = %d", soma);

    return 0;
}
```

# Problema

8

- Calcular o fatorial de quatro números e somá-los:
  - $2! + 3! + 4! + 5! = 152$ ;

MESMAS OPERAÇÕES SENDO  
UTILIZADAS VÁRIAS VEZES

```
int main(void)
{
    int fat1, fat2, fat3, fat4, num1, num2, num3, num4;
    int soma, i;

    printf("Informe um valor: ");
    scanf("%d", &num1);
    printf("Informe um valor: ");
    scanf("%d", &num2);
    printf("Informe um valor: ");
    scanf("%d", &num3);
    printf("Informe um valor: ");
    scanf("%d", &num4);

    //FAT DO PRIMEIRO NUMERO
    fat1 = 1;
    for(i = num1; i > 0 ; i--)
        fat1 *= i;

    //FAT DO SEGUNDO NUMERO
    fat2 = 1;
    for(i = num2; i > 0 ; i--)
        fat2 *= i;

    //FAT DO TERCEIRO NUMERO
    fat3 = 1;
    for(i = num3; i > 0 ; i--)
        fat3 *= i;

    //FAT DO QUARTO NUMERO
    fat4 = 1;
    for(i = num4; i > 0 ; i--)
        fat4 *= i;

    soma = fat1 + fat2 + fat3 + fat4;
    printf("soma = %d", soma);

    return 0;
}
```



# Problema

9

- Calcular o fatorial de quatro números e somá-los.
  - ▣  $2! + 3! + 4! + 5! = 152$ ;

## O QUE FAZER PARA REUTILIZAR A MESMA OPERAÇÃO?

MESMAS OPERAÇÕES SENDO UTILIZADAS VÁRIAS VEZES

```
int main(void)
{
    int fat1, fat2, fat3, fat4, num1, num2, num3, num4;
    int soma, i;

    printf("Informe um valor: ");
    scanf("%d", &num1);
    printf("Informe um valor: ");
    scanf("%d", &num2);
    printf("Informe um valor: ");
    scanf("%d", &num3);
    printf("Informe um valor: ");
    scanf("%d", &num4);

    for(i = num2; i > 0 ; i--)
        fat2 *= i;
    //FAT DO TERCEIRO NUMERO
    fat3 = 1;
    for(i = num3; i > 0 ; i--)
        fat3 *= i;
    //FAT DO QUART NUMERO
    fat4 = 1;
    for(i = num4; i > 0 ; i--)
        fat4 *= i;

    soma = fat1 + fat2 + fat3 + fat4;
    printf("soma = %d", soma);

    return 0;
}
```

# Funções

10

- Nomeação de um conjunto de comandos agrupados, que podem ser utilizados várias vezes dentro do mesmo código.
  
- Vantagens:
  - ▣ Auxiliar na estrutura do código;
  - ▣ Reutilização de código.

# Funções

11

- Exemplo de função em C:
  - `scanf("%d", &x);`
  - `printf("%lf", dado);`

# Funções

12

- Exemplo de função em C:
  - ▣ `scanf("%d", &x);`
  - ▣ `printf("%lf", dado);`
  
- Código que realiza as operações está na biblioteca `<stdio.h>`.

# Funções

13

- Exemplo de função em C:
  - `scanf("%d", &x);`
  - `printf("%lf", dado);`
- Código que realiza as operações está na biblioteca `<stdio.h>`.
- A função é chamada e executa de acordo com o que lhe é passado.

# Funções

14

- Possibilita reutilizar trechos de código.
- Traz modularização, segmentação do código.
- São os **subprogramas**.
- Sinônimos: procedimento, função, módulo, método, subrotina e componente.

# Funções

15

- Divididas em:
  - ▣ 1 – cabeçalho.
  - ▣ 2 – corpo ou conjunto de instruções.

(1) `<tipo> <nome> (<parâmetros>)`

(2) `{  
    <corpo ou instruções>  
}`

# Funções

16

- **Tipo**: define o tipo de dado do retorno da função.
  - void para função sem retorno.

```
tipo nome (parâmetros)
```



# Funções

17

- **Nome:** identificador ou nome atribuído à função.
  - ▣ A função é chamada a partir do nome.

```
tipo nome (parâmetros)
```

# Funções

18

- **Parâmetros:** permite ao programador passar informação de um trecho de código para dentro da função. São os dados de entrada.
- void quando a função não possui parâmetros.

```
tipo nome (parâmetros)
```

# Funções

19

- **Corpo:** processa as entradas (parâmetros) e gera a saída (return).
  - ▣ Utilizam declarações e comandos.

```
{  
    <corpo ou instruções>  
}
```

# Criando uma função

20

- Parâmetros de entrada:
- Procedimento:
- Saída:

# Criando uma função

21

- Parâmetros de entrada:
  - ▣ Lista de parâmetros da função;
  - ▣ Valores de entrada que serão trabalhados;
  - ▣ Variáveis generalizadas.
- Procedimento:
- Saída:

# Criando uma função

22

- Parâmetros de entrada:
  - ▣ Lista de parâmetros da função;
  - ▣ Valores de entrada que serão trabalhados;
  - ▣ Variáveis generalizadas.
- Procedimento:
  - ▣ Comandos no corpo da função;
  - ▣ Transformar entrada em saída.
- Saída:

# Criando uma função

23

- Parâmetros de entrada:
  - ▣ Lista de parâmetros da função;
  - ▣ Valores de entrada que serão trabalhados;
  - ▣ Variáveis generalizadas.
- Procedimento:
  - ▣ Comandos no corpo da função;
  - ▣ Transformar entrada em saída.
- Saída:
  - ▣ Impressão em tela;
  - ▣ Retorno (return).

# Funções: Exemplo

24

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}

int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```



# Funções

25

FUNÇÃO {

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

# Funções

26

FUNÇÃO {

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

TIPO

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

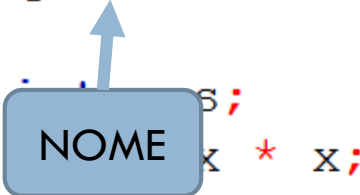
    return 0;
}
```

# Funções

27

FUNÇÃO {

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```



```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

# Funções

28

FUNÇÃO {

```
int quadrado(int x)
{
    int res
    return res;
}
```

PARÂMETROS

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

# Funções

29

FUNÇÃO {

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

CORPO

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

# Funções

30

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

FUNÇÃO  
PRINCIPAL

```
{
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

# Funções

31

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

FUNÇÃO  
PRINCIPAL

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

CHAMADA DA  
FUNÇÃO

A blue arrow points from the text box 'CHAMADA DA FUNÇÃO' to the line 'resultado = quadrado(numero);' in the main function code block.

# Funções

32

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

FUNÇÃO  
PRINCIPAL

```
int main(void)
{
    int numero = 3;
    int resultado;
    resultado = quadrado(numero);
    printf("%d^2 = %d", numero, resultado);
    return 0;
}
```

RECEBER  
RETORNO



# Funções

33

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

**MEMÓRIA**

# Funções

34

**TODO PROGRAMA  
SEMPRE COMEÇA  
EXECUTANDO NO INÍCIO  
DA MAIN**

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

**MEMÓRIA**

# Funções

35

cria variável na  
memória

```
int main(void)
```

```
{
```

```
→ int numero = 3;
```

```
    int resultado;
```

```
    resultado = quadrado(numero);
```

```
    printf("%d^2 = %d", numero, resultado);
```

```
    return 0;
```

```
}
```

MEMÓRIA

numero

# Funções

36

ARMAZENA VALOR NA  
VARIÁVEL CRIADA

```
int main(void)
```

```
{
```

```
→ int numero = 3;  
   int resultado;
```

```
    resultado = quadrado(numero);
```

```
    printf("%d^2 = %d", numero, resultado);
```

```
    return 0;
```

```
}
```

MEMÓRIA

numero	3
--------	---

# Funções

37

CRIA VARIÁVEL NA  
MEMÓRIA

```
int main(void)
{
    int numero = 3;
    → int resultado;
```

```
    resultado = quadrado(numero);
```

```
    printf("%d^2 = %d", numero, resultado);
```

```
    return 0;
```

```
}
```

MEMÓRIA

numero 3

resultado

# Funções

38

ESPERA O RESULTADO QUE  
SERÁ RETORNADO DA  
FUNÇÃO

```
int main(void)
{
```

```
    int numero = 3;
    int resultado;
```

```
    → resultado = quadrado(numero);
```

```
    printf("%d^2 = %d", numero, resultado);
```

```
    return 0;
```

```
}
```

## MEMÓRIA

numero	3
--------	---

resultado	
-----------	--

# Funções

39

CHAMADA DA FUNÇÃO.  
PROGRAMA FICA NESSE  
PONTO ATÉ FUNÇÃO  
FINALIZAR.

```
int main(void)
{
```

```
    int numero = 3;
    int resultado;
```

```
    → resultado = quadrado(numero);
```

```
    printf("%d^2 = %d", numero, resultado);
```

```
    return 0;
```

```
}
```


## MEMÓRIA

numero	3
--------	---

resultado	
-----------	--

# Funções

40



```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

INICIA FUNÇÃO A PARTIR  
DO NOME

MEMÓRIA

numero	3
--------	---

resultado	
-----------	--

```
→ resultado = quadrado(numero);
printf("%d^2 = %d", numero, resultado);

return 0;
}
```



# Funções

41

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

cria variável na  
memória...

## MEMÓRIA

numero	3
--------	---

resultado	
-----------	--

x	
---	--

→ resultado = quadrado(numero);


```
printf("%d^2 = %d", numero, resultado);
```

```
return 0;
```

```
}
```

# Funções

42



```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

cria variável na  
memória e recebe cópia  
do valor passado na  
chamada

## MEMÓRIA



```
→ resultado = quadrado(numero);
printf("%d^2 = %d", numero, resultado);

return 0;
}
```

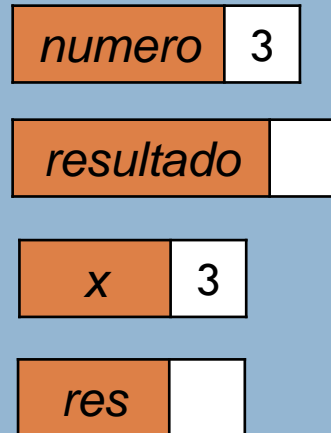
# Funções

43

```
int quadrado(int x)
{
    → int res;
    res = x * x;
    return res;
}
```

CRIA VARIÁVEL NA  
MEMÓRIA

## MEMÓRIA



```
→ resultado = quadrado(numero);

printf("%d^2 = %d", numero, resultado);

return 0;
}
```

# Funções

44

```
int quadrado(int x)
{
    int res;
    → res = x * x;
    return res;
}
```

REALIZA CÁLCULO E  
ARMAZENA NA VARIÁVEL

→ resultado = quadrado(numero);

```
printf("%d^2 = %d", numero, resultado);
```

```
return 0;
```

```
}
```

## MEMÓRIA

numero	3
--------	---

resultado	
-----------	--

x	3
---	---

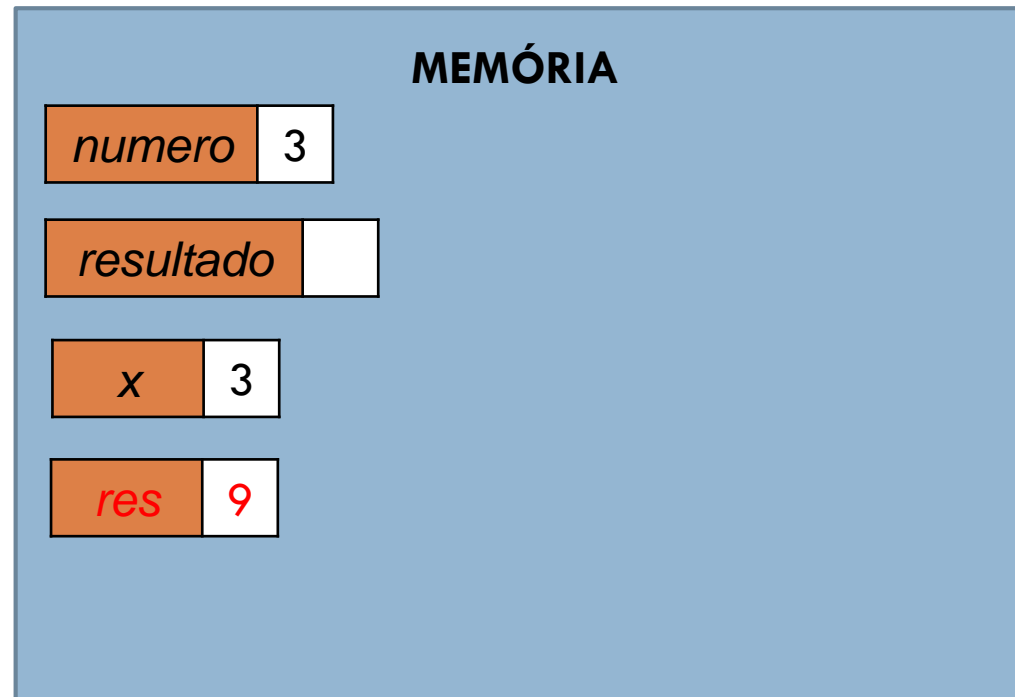
res	9
-----	---

# Funções

45

```
int quadrado(int x)
{
    int res;
    res = x * x;
    → return res;
}
```

**RETORNA VALOR DA  
VARIÁVEL PARA QUEM  
CHAMOU A FUNÇÃO**



```
→ resultado = quadrado(numero);

printf("%d^2 = %d", numero, resultado);

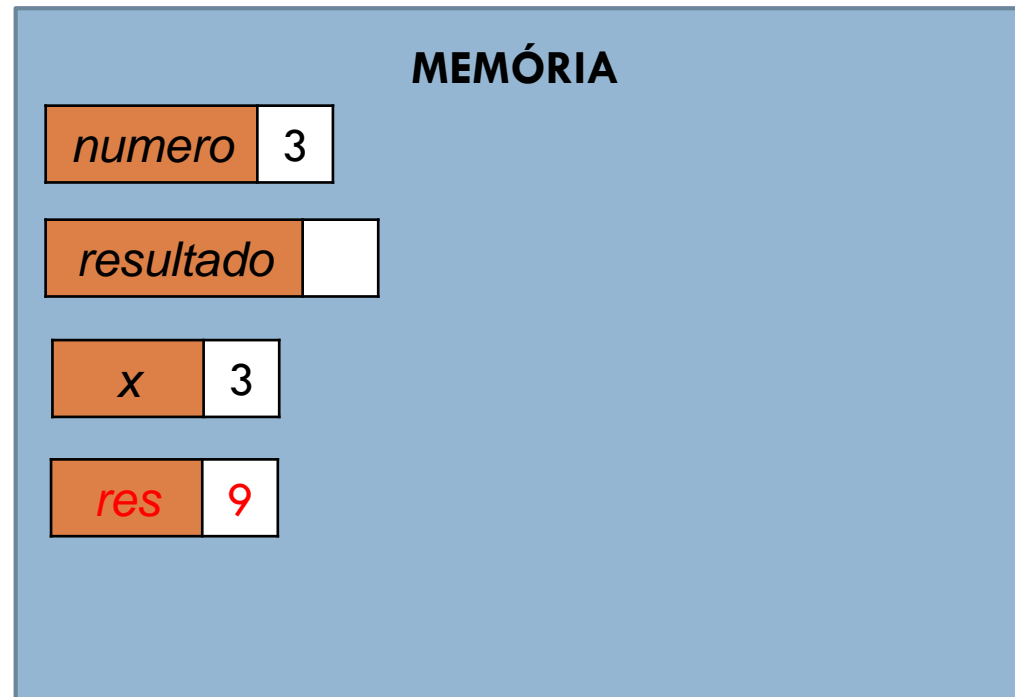
return 0;
}
```

# Funções

46

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

**VOLTA A EXECUTAR A  
PARTIR DE ONDE A  
FUNÇÃO FOI CHAMADA**



```
→ resultado = quadrado(numero);
printf("%d^2 = %d", numero, resultado);

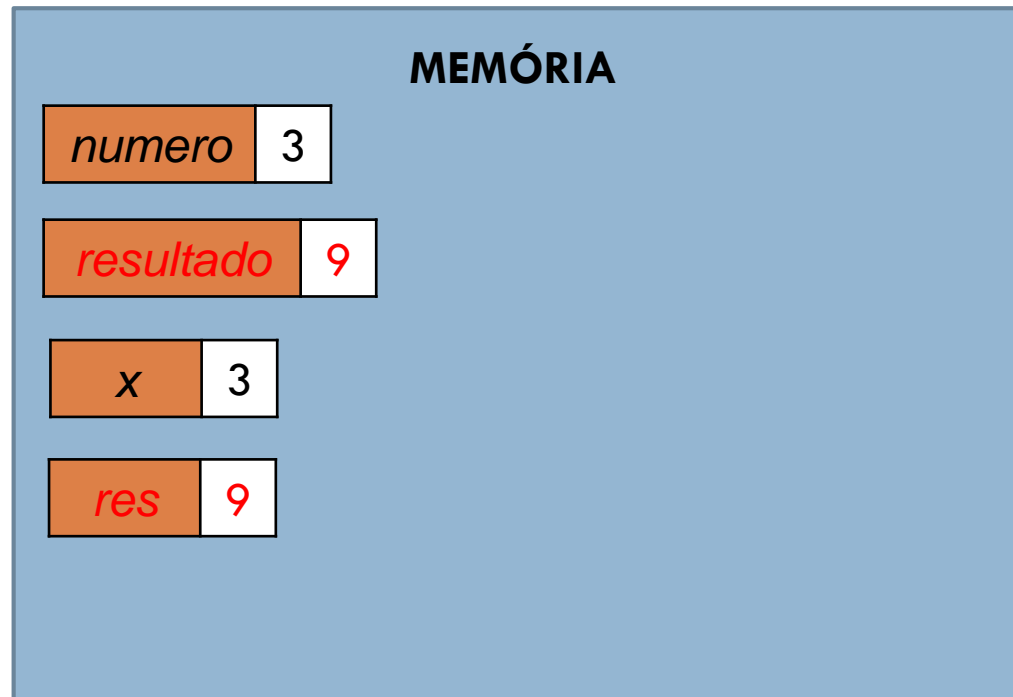
return 0;
}
```

# Funções

47

```
int quadrado(int x)
{
    int res;
    res = x * x;
    return res;
}
```

**VARIÁVEL RECEBE O  
RETORNO DA FUNÇÃO**



```
→ resultado = quadrado(numero);
printf("%d^2 = %d", numero, resultado);

return 0;
}
```

# Funções

48

IMPRIMIR VALORES NA TELA

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    → printf("%d^2 = %d", numero, resultado);

    return 0;
}
```

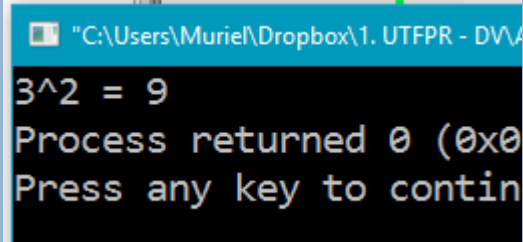
## MEMÓRIA

numero	3
--------	---

resultado	9
-----------	---

x	3
---	---

res	9
-----	---



```
"C:\Users\MurIEL\Dropbox\1. UTFPR - DVV\
3^2 = 9
Process returned 0 (0x0)
Press any key to continue
```



# Funções

49

FINALIZA O PROGRAMA

```
int main(void)
{
    int numero = 3;
    int resultado;

    resultado = quadrado(numero);

    printf("%d^2 = %d", numero, resultado);

    → return 0;
}
```

## MEMÓRIA

numero	3
--------	---

resultado	9
-----------	---

x	3
---	---

res	9
-----	---

# Cabeçalho da função

50

- Declara quantos parâmetros forem necessário;

# Cabeçalho da função

51

- Declara quantos parâmetros forem necessário;
  - ▣ **Definir o tipo de cada parâmetro**, separados por vírgula.

# Cabeçalho da função

52

- Declara quantos parâmetros forem necessário;
  - ▣ **Definir o tipo de cada parâmetro**, separados por vírgula.
- Apenas um retorno;

# Cabeçalho da função

53

- Declara quantos parâmetros forem necessário;
  - ▣ **Definir o tipo de cada parâmetro**, separados por vírgula.
- Apenas um retorno;
  - ▣ **Definir o tipo de dado** que será retornado com o operador *return*.

# Cabeçalho da função

54

- Declara quantos parâmetros forem necessário;
  - ▣ Definir o tipo de cada parâmetro, separados por vírgula.
- Apenas um retorno;
  - ▣ Definir o tipo de dado que será retornado com o operador *return*.

```
float raiz(float r)
double hipotenusa(double c, double b)
void imprimir(int RA, char Letra, double peso)

double hipotenusa(double c, b) //ERRADO
```

# Regras de implementação

55

- Primeira regra:
  - ▣ Se as funções forem implementadas **acima da main,** elas **devem estar em ordem.**

# Regras de implementação

56

- Primeira regra:
  - ▣ Se as funções forem implementadas **acima da main**, elas **devem estar em ordem**.

```
#include <stdio.h>

int multiplica(int a, int b)
{
    return a * b;
}

int quadrado(int x)
{
    int res;
    res = multiplica(x, x);
    return res;
}

int main(void)
{
    int numero = 3;
    int resultado;
    resultado = quadrado(numero);
    printf("%d^2 = %d", numero, resultado);
    return 0;
}
```



# Regras de implementação

57

- Primeira regra:
  - ▣ Se as funções forem implementadas **acima da main**, elas **devem estar em ordem**.
  - ▣ Função **chamada** sempre acima da **chamadora**.

```
#include <stdio.h>

int multiplica(int a, int b)
{
    return a * b;
}

int quadrado(int x)
{
    int res;
    res = multiplica(x, x);
    return res;
}

int main(void)
{
    int numero = 3;
    int resultado;
    resultado = quadrado(numero);
    printf("%d^2 = %d", numero, resultado);
    return 0;
}
```

FUNÇÃO CHAMADA

FUNÇÃO CHAMADORA

# Regras de implementação

58

- Segunda regra:
  - ▣ Se as funções forem implementadas **fora de ordem**, é necessário **incluir os protótipos** logo após a inclusão de bibliotecas.

# Regras de implementação

59

- Segunda regra:
  - ▣ Se as funções forem implementadas **fora de ordem**, é necessário **incluir os protótipos** logo após a inclusão de bibliotecas.

```
#include <stdio.h>

int multiplica(int a, int b);
int quadrado(int x);

int main(void)
{
    int numero = 3;
    int resultado;
    resultado = quadrado(numero);
    printf("%d^2 = %d", numero, resultado);
    return 0;
}

int quadrado(int x)
{
    int res;
    res = multiplica(x, x);
    return res;
}

int multiplica(int a, int b)
{
    return a * b;
}
```

# Regras de implementação

60

- Segunda regra:
  - ▣ Se as funções forem implementadas **fora de ordem**, é necessário **incluir os protótipos** logo após a inclusão de bibliotecas.
  - ▣ **Protótipos** são os cabeçalhos seguidos de ponto e virgula.

```
#include <stdio.h>

int multiplica(int a, int b);
int quadrado(int x);

int main(void)
{
    int numero = 3;
    int resultado;
    resultado = quadrado(numero);
    printf("%d^2 = %d", numero, resultado);
    return 0;
}

int quadrado(int x)
{
    int res;
    res = multiplica(x, x);
    return res;
}

int multiplica(int a, int b)
{
    return a * b;
}
```

PROTÓTIPOS

# Regras de implementação

61

- Terceira regra:
  - Os parâmetros  
**DEVEM** ser  
**passados** e **DEVEM**  
**seguir a ordem e**  
**tipo** do protótipo  
(cabeçalho).

# Regras de implementação

62

- Terceira regra:
  - ▣ Os parâmetros **DEVEM** ser **passados** e **DEVEM** seguir a ordem e tipo do protótipo (cabeçalho).

```
#include <stdio.h>

int multiplica(int a, int b);
int quadrado(int x);
float raiz(float r);
void imprimir(int RA, char Letra, double peso);

int main(void)
{
    int var1 = 2, res;
    float resultado;

    //passar valores
    res1 = multiplica(5,3);
    //passar valores de variaveis
    res = quadrado(res1);
    //passar o retorno da função como parametro
    resultado = raiz(multiplica(var1, 5));
    //função sem retorno
    imprimir(137, 'M', 74.6);
    //passar como parametro para funções
    printf("%d", quadrado(var1));

    return 0;
}
```

# Regras de implementação

63

- Terceira regra:
  - ▣ Os parâmetros **DEVEM** ser **passados** e **DEVEM** seguir a ordem e tipo do protótipo (cabeçalho).
  - ▣ As funções são executadas da mais **interna** para a mais **externa**.

```
#include <stdio.h>

int multiplica(int a, int b);
int quadrado(int x);
float raiz(float r);
void imprimir(int RA, char Letra, double peso);

int main(void)
{
    int var1 = 2, res;
    float resultado;

    //passar valores
    res1 = multiplica(5,3);
    //passar valores de variaveis
    res = quadrado(res1);
    //passar o retorno da função como parametro
    resultado = raiz(multiplica(var1, 5));
    //função sem retorno
    imprimir(137, 'M', 74.6);
    //passar como parametro para funções
    printf("%d", quadrado(var1));

    return 0;
}
```

# Exemplos de funções

64

```
#include <stdio.h>
int menu(void);

int main(void)
{
    int op = menu();
    printf("\n\nPrimeira Opcao: %d\n", op);
    op = menu();
    printf("\n\nSegunda Opcao: %d\n", op);
    return 0;
}

int menu(void)
{
    int opcao;
    do
    {
        printf("Escolha uma opcao:\n");
        printf("(1) Opcao 1;\n");
        printf("(2) Opcao 2;\n");
        printf("(3) Opcao 3;\n");
        scanf("%d", &opcao);

    }while(opcao < 1 || opcao > 3);

    return opcao;
}
```



# Exemplos de funções

65

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

```
int fatorial(int NUM)
{
    int i, fat = 1;
    for(i = NUM; i > 0 ; i--)
        fat *= i;

    return fat;
}
```

```
int main(void)
{
    int num1, num2, num3, num4;
    int soma, i;

    printf("Informe um valor: ");
    scanf("%d", &num1);
    printf("Informe um valor: ");
    scanf("%d", &num2);
    printf("Informe um valor: ");
    scanf("%d", &num3);
    printf("Informe um valor: ");
    scanf("%d", &num4);

    soma = fatorial(num1);
    soma += fatorial(num2);
    soma += fatorial(num3);
    soma += fatorial(num4);
    printf("soma = %d", soma);

    return 0;
}
```

# Questionário

66

- 1) Defina o que é função e qual é a sua utilidade.
- 2) Escreva uma função de exemplo e descreva sua estrutura.

# Questionário

67

- 3) Escreva uma função em C que receba dois valores double e retorne qual é maior.
- 4) Escreva uma função em C que receba dois valores double e retorne qual é menor.
- 5) Escreva uma função main que utilize as funções anteriores.

# Questionário

68

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

```
double max(double A, double B)
{
    if(A > B) return A;
    else return B;
}
```

```
double min(double A, double B)
{
    if(A < B) return A;
    else return B;
}
```

```
int main(void)
{
```

```
    double X = 10.59;
    double Y = 90.789;
```

```
    printf("Max: %lf\n", max(X,Y));
    printf("Min: %lf\n", min(X,Y));
```

```
    return 0;
```

```
}
```

# Exemplo de função

69

- Função para imprimir os divisores de um número.
  - ▣ Parâmetros (entrada):
  - ▣ Processamento (corpo):
  - ▣ Saída (return):

# Exemplo de função

70

- Função para imprimir os divisores de um número.
  - ▣ Parâmetros (entrada): um número inteiro.
  - ▣ Processamento (corpo): varrer todos os valores entre 1 e o número, verificando quais geram resto zero.
  - ▣ Saída (return): sem retorno, apenas impressão dos divisores.

# Exemplo de função

71

## □ Função

### ▣ Parâmetro

### ▣ Processo

e o número

### ▣ Saída

divisores

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

```
void divisores(int num)
```

```
{
```

```
    int i = 1;
```

```
    printf("Divisores de %d: ", num);
```

```
    for(i = 1; i <= num; i++)
```

```
    {
```

```
        if( (num%i) == 0 ) printf("%d ", i);
```

```
    }
```

```
    printf("\n\n");
```

```
}
```

```
int main(void)
```

```
{
```

```
    int valor = 10;
```

```
    divisores(valor);
```

```
    divisores(50);
```

```
    divisores(35);
```

```
    valor = 7;
```

```
    divisores(valor);
```

```
    return 0;
```

```
}
```

número.

divisores entre 1

zero.

saída dos