

Linguagem C: Funções

Sumário

- Funções:
 - Funções da biblioteca matemática;
- Definições de funções;
- O comando return;
- Protótipos de funções;
- Regras de escopos:
 - Variáveis locais;
 - Parâmetros formais;
 - Variáveis globais;

Sumário

- Parâmetros de funções;
- Passagem de parâmetros:
 - Passagem por valor;
 - Passagem por referência;

Funções

- A maioria dos programas destinados a resolver os problemas do mundo real são **muito maiores** do que os programas desenvolvidos por nós até o momento;
- A experiência tem mostrado que a melhor maneira de desenvolver e manter um programa grande é contruí-lo a partir de **pequenas partes ou componentes**;
- Esses componentes ou módulos são chamados de **funções**;

Funções

- Uma função em C é portanto, uma sub-rotina que contém um ou mais comandos em C e que executa(m) uma ou mais tarefas;
- Uma função pode ser definida pelo programador com algumas finalidades:
 - ❑ **Reuso:** definir tarefas específicas que podem ser usadas em muitos pontos do programa;
 - ❑ **Legibilidade:** evita a exposição de códigos (caixa-preta) facilitando a legibilidade do código;
 - ❑ **Modularidade:** a divisão do código em partes facilita a identificação de erros por parte do programador;
 - ❑ **Portabilidade:** funções não fazem parte do conjunto básico da linguagem, evitando problemas de suporte aos diversos padrões de vídeo, teclados, S.Os, etc

Funções da biblioteca matemática

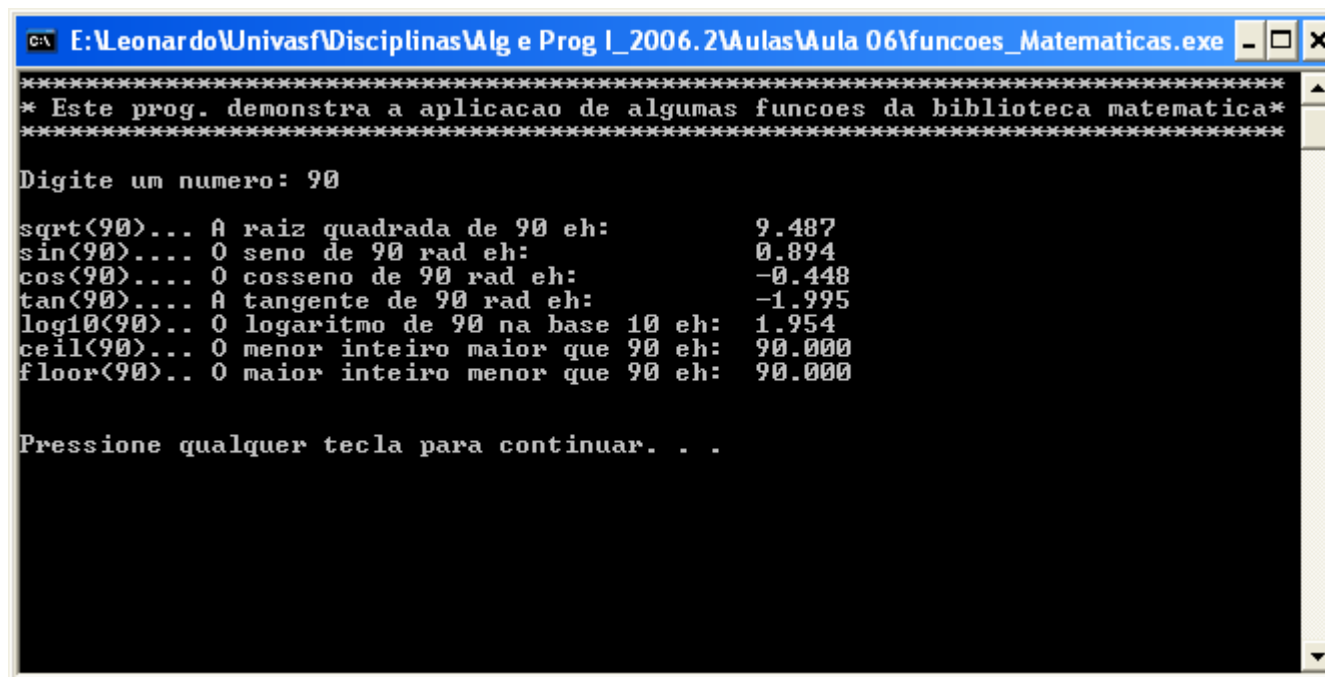
- O melhor exemplo de como atuam as funções é através das funções da biblioteca matemática, vejamos:

```
#include <stdio.h>
#include <math.h>
/* Este programa inicia o conceito de funções através da biblioteca matemática */
int main()
{
    int num;
    printf("*****");
    printf("\n* Este prog. demonstra a aplicacao de algumas funcoes da biblioteca matematica*");
    printf("\n*****");
    printf("\n\nDigite um numero: ");
    scanf("%d", &num);
    printf("\nsqrt(%d)... A raiz quadrada de %d eh:      %.3f", num, num, sqrt(num));
    printf("\nsin(%d)... O seno de %d rad eh:              %.3f", num, num, sin(num));
    printf("\ncos(%d)... O cosseno de %d rad eh:           %.3f", num, num, cos(num));
    printf("\ntan(%d)... A tangente de %d rad eh:            %.3f", num, num, tan(num));
    printf("\nlog10(%d).. O logaritmo de %d na base 10 eh:    %.3f", num, num, log10(num));
    printf("\nceil(%d)... O menor inteiro maior que %d eh:    %.3f", num, num, ceil(num));
    printf("\nfloor(%d).. O maior inteiro menor que %d eh:      %.3f\n\n\n", num, num, floor(num));
    system("pause");
    return 0;
}
```

Chamadas às funções

Funções da biblioteca matemática

- A saída no console para o programa anterior será:



```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_Matematicas.exe
*****
* Este prog. demonstra a aplicacao de algumas funcoes da biblioteca matematica *
*****

Digite um numero: 90

sqrt(90)... A raiz quadrada de 90 eh:          9.487
sin(90).... 0 seno de 90 rad eh:              0.894
cos(90).... 0 cosseno de 90 rad eh:           -0.448
tan(90).... A tangente de 90 rad eh:          -1.995
log10(90).. 0 logaritmo de 90 na base 10 eh:   1.954
ceil(90)... 0 menor inteiro maior que 90 eh:  90.000
floor(90).. 0 maior inteiro menor que 90 eh:   90.000

Pressione qualquer tecla para continuar. . .
```

Definições de Funções

- Todo programa apresentado até aqui consiste em uma função denominada **main** que chamou as funções da biblioteca padrão para realizar suas tarefas;
- Vejamos como os programadores podem escrever suas próprias funções personalizadas:

tipo_de_retorno nome_da_funcao (parametros)

{

Tipo da variável que a função irá retornar

corpo_da_funcao;

Mesma especificações de nomes de uma variável

Lista de "variáveis" (tipo nome1, tipo nome2,..., tipo nomeX)

}

Definições de Funções

- Considere um programa que usa a função celsius para converter uma temperatura digitada em fahrenheit para graus celsius, vejamos:

```
#include<stdio.h>
/* Este programa converte uma temperatura dada em Fahrenheit para graus Celsius */
int celsius(int fahr)
{
    int c;
    c = (fahr - 32) * 5/9;
    return c;
}
int main()
{
    int c, f;
    printf("*****");
    printf("\n* Este programa converte uma temperatura de Fahrenheit para Celsius *");
    printf("\n*****");
    printf("\n\nDigite a temperatura em graus Fahrenheit: ");
    scanf("%d", &f);
    c = celsius(f);
    printf("\nA temperatura %d F em Celsius eh: %d C\n\n", f, c);
    system("pause");
    return 0;
}
```

Tipo de retorno da função

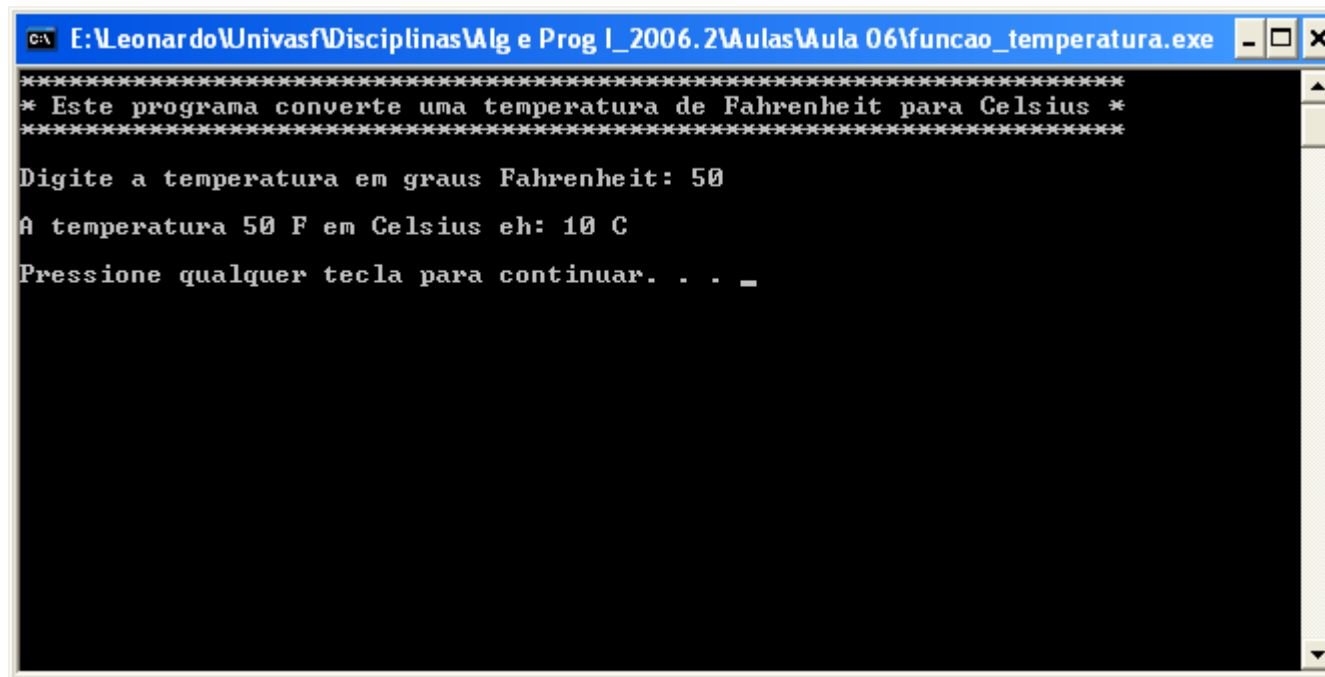
Nome da Função

Definição da Função

Parâmetro ou argumento da função

Definições de Funções

- A saída no console para o programa anterior será:

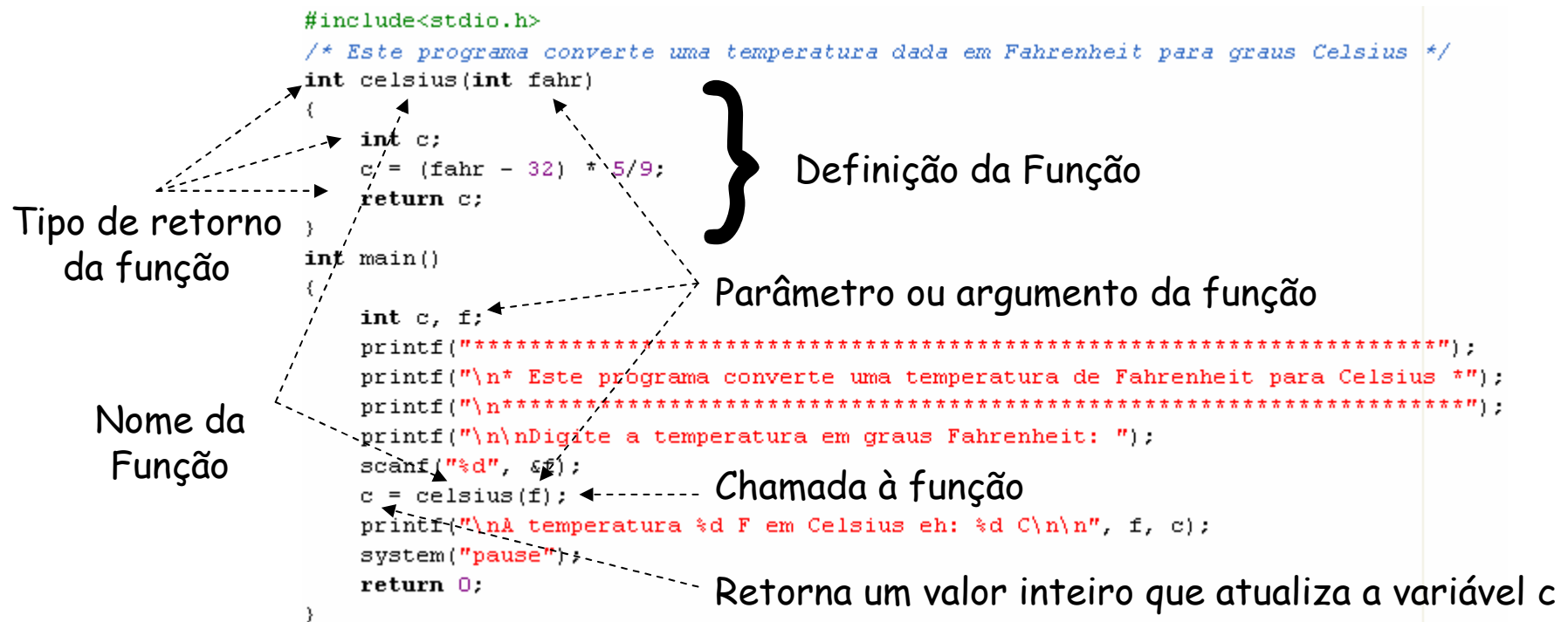


```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcao_temperatura.exe
*****
* Este programa converte uma temperatura de Fahrenheit para Celsius *
*****

Digite a temperatura em graus Fahrenheit: 50
A temperatura 50 F em Celsius eh: 10 C
Pressione qualquer tecla para continuar. . . _
```

Definições de Funções

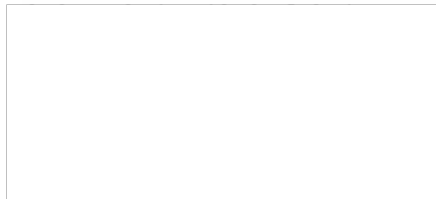
- Analisemos agora a relação da função celsius com a função main, vejamos:



Definições de Funções

- Podemos entender melhor a analogia feita de uma função a uma **caixa-preta**;

```
#include<stdio.h>
/* Este programa converte uma temperatura dada em Fahrenheit para graus Celsius */
```



Não importa como mas... Qualquer temperatura em Fahrenheit passada à função retornará essa temperatura convertida para graus celsius

```
int main()
{
    int c, f;
    printf("*****");
    printf("\n* Este programa converte uma temperatura de Fahrenheit para Celsius *");
    printf("\n*****");
    printf("\n\nDigite a temperatura em graus Fahrenheit: ");
    scanf("%d", &f);
    c = celsius(f);
    printf("\nA temperatura %d F em Celsius eh: %d C\n\n", f, c);
    system("pause");
    return 0;
}
```

O comando return

- O comando return **termina a execução da função** e retorna o controle para a instrução seguinte do código de chamada;
- A sintaxe de uma instrução return tem uma das **três formas** seguintes:

return;

Somente finaliza a função. O mesmo que uma função do tipo **void**.

return expressão;

return (expressão);

O valor é retornado à função que chama. Este valor é convertido para o tipo da função

O comando return

- Portanto, podemos eliminar a variável desnecessária declarada no corpo da função celsius e colocar uma expressão diretamente, vejamos:

```
#include<stdio.h>
/* Este programa converte uma temperatura dada em Fahrenheit para graus Celsius */
int celsius(int fahr)
{
    return (fahr - 32) * 5/9; ←----- Cálculo junto ao comando
                                return diretamente.
}
int main()
{
    int c, f;
    printf("*****");
    printf("\n* Este programa converte uma temperatura de Fahrenheit para Celsius *");
    printf("\n*****");
    printf("\n\nDigite a temperatura em graus Fahrenheit: ");
    scanf("%d", &f);
    c = celsius(f);
    printf("\nA temperatura %d F em Celsius eh: %d C\n\n", f, c);
    system("pause");
    return 0;
}
```

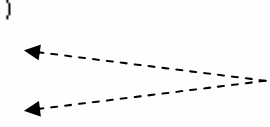
O comando return

- O comando return pode retornar somente **um único valor** para a função que chama;

- Isso não impede de ter vários return em uma função, vejamos;

```
#include<stdio.h>
/*Este prog mostra a utilização de 2 comandos return numa função*/
int condicao(char teste)
{
    if(teste<='Z')
        return 1;
    else
        return 0;
}

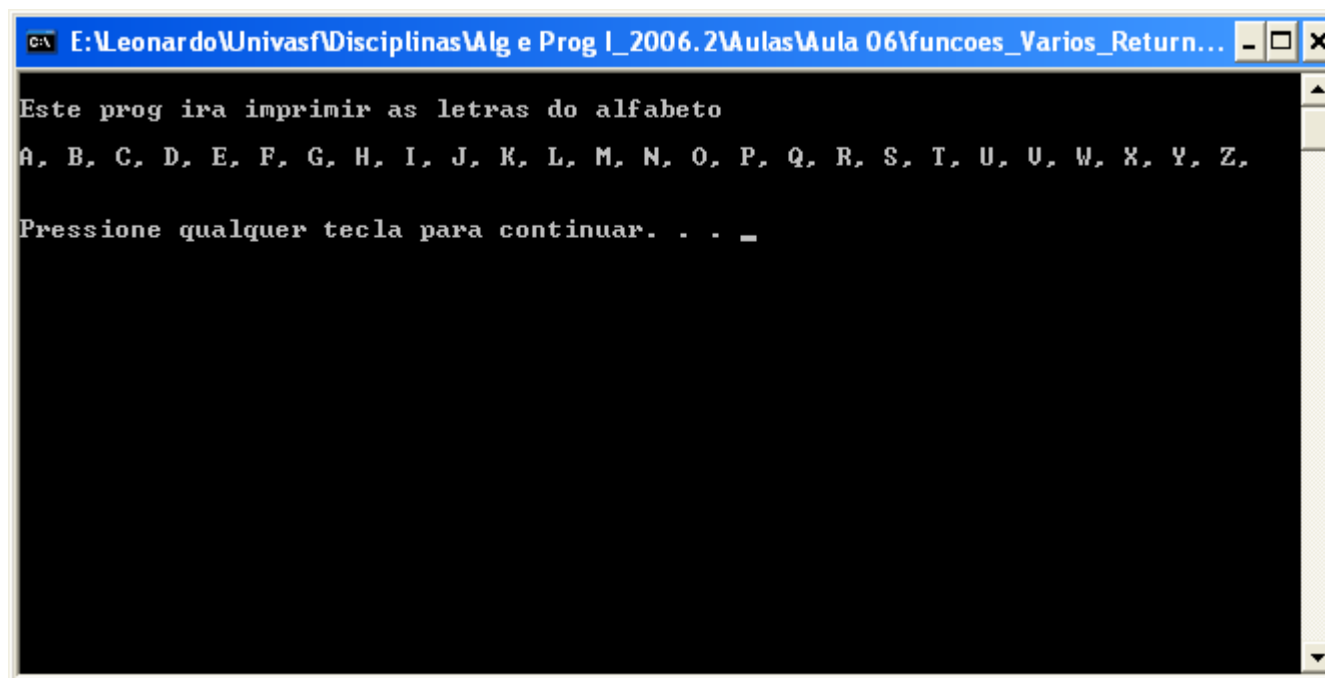
int main()
{
    char letra = 'A';
    printf("\nEste prog ira imprimir as letras do alfabeto\n\n");
    do
    {
        printf("%c, ", letra);
        letra++;
    } while( condicao(letra) );
    printf("\n\n\n");
    system("pause");
    return 0;
}
```



Apesar de ter mais de um return apenas um será executado por vez

O comando return

- A saída no console para o programa anterior será:



A screenshot of a Windows command prompt window. The title bar at the top reads "E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_Varios_Return...". The window has a black background with white text. The text displayed is: "Este prog ira imprimir as letras do alfabeto", followed by a line of letters "A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z," on the next line, and "Pressione qualquer tecla para continuar. . . _" on the third line. The cursor is positioned at the end of the third line.

```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_Varios_Return...
Este prog ira imprimir as letras do alfabeto
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z,
Pressione qualquer tecla para continuar. . . _
```


Protótipos de funções

- Até o momento, as funções apareceram antes da função `main()`;
- Assim como ocorre com uma variável, não podemos **usar** uma função **sem antes declará-la**;
- A declaração de uma função é chamada **protótipo** e é uma instrução que **estabelece o tipo** da função e dos argumentos que ela recebe;

Protótipos de funções

- Vejamos nosso exemplo com o protótipo da função celsius;

```
#include<stdio.h>
/* Este programa converte uma temperatura dada em Fahrenheit para graus Celsius */
int celsius(int fahr); ←----- Protótipo da função celsius
int main()
{
    int c, f;
    printf("*****");
    printf("\n* Este programa converte uma temperatura de Fahrenheit para Celsius *");
    printf("\n*****");
    printf("\n\nDigite a temperatura em graus Fahrenheit: ");
    scanf("%d", &f);
    c = celsius(f);
    printf("\nA temperatura %d F em Celsius eh: %d C\n\n", f, c);
    system("pause");
    return 0;
}
int celsius(int fahr) ←----- Definição da função após a
{                               função main( )
    return (fahr - 32) * 5/9;
}
```

Protótipos de funções

- Poderíamos então transformar nossa função principal em chamadas sucessivas de funções, vejamos:

```
#include<stdio.h>
/* Este programa converte uma temperatura dada em Fahrenheit para graus Celsius */
void apresentacao();
int setFahr();
int celsius(int fahr);
void printResult();
int main()
{
    int c, f;
    apresentacao();
    f = setFahr();
    c = celsius(f);
    printResult(c, f);
    return 0;
}
...
```

Protótipos de funções

- As definições das funções do programa anterior são:

```
void apresentacao()
{
    printf("*****");
    printf("\n* Este programa converte uma temperatura de Fahrenheit para Celsius *");
    printf("\n*****");
}

int setFahr()
{
    int f;
    printf("\n\nDigite a temperatura em graus Fahrenheit: ");
    scanf("%d", &f);
    return f;
}

int celsius(int fahr)
{
    return (fahr - 32) * 5/9;
}

void printResult(int c, int f)
{
    printf("\nA temperatura %d F em Celsius eh: %d C\n\n", f, c);
    system("pause");
}
```

Exemplo

- Escreva a função que dará sentido à calculadora proposta pelo programa abaixo:

```
#include <stdio.h>
/* Este programa executa as quatro operações básicas aritméticas */
```

```
void calculadora(float a, float b, char op);
```

```
int main()
{
```

```
    float op1, op2;
```

```
    char operacao;
```

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

```
    printf("\n* Este programa executa as quatro operacoes basicas aritmeticas *");
```

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

```
    printf("\n\nDigite o primeiro operando: ");
```

```
    scanf("%f", &op1);
```

```
    printf("\nDigite o segundo operando: ");
```

```
    scanf("%f", &op2);
```

```
    printf("\nDigite o operador: ");
```

```
    operacao = getche();
```

```
    calculadora(op1, op2, operacao);
```

```
    system("pause");
```

```
    return 0;
```

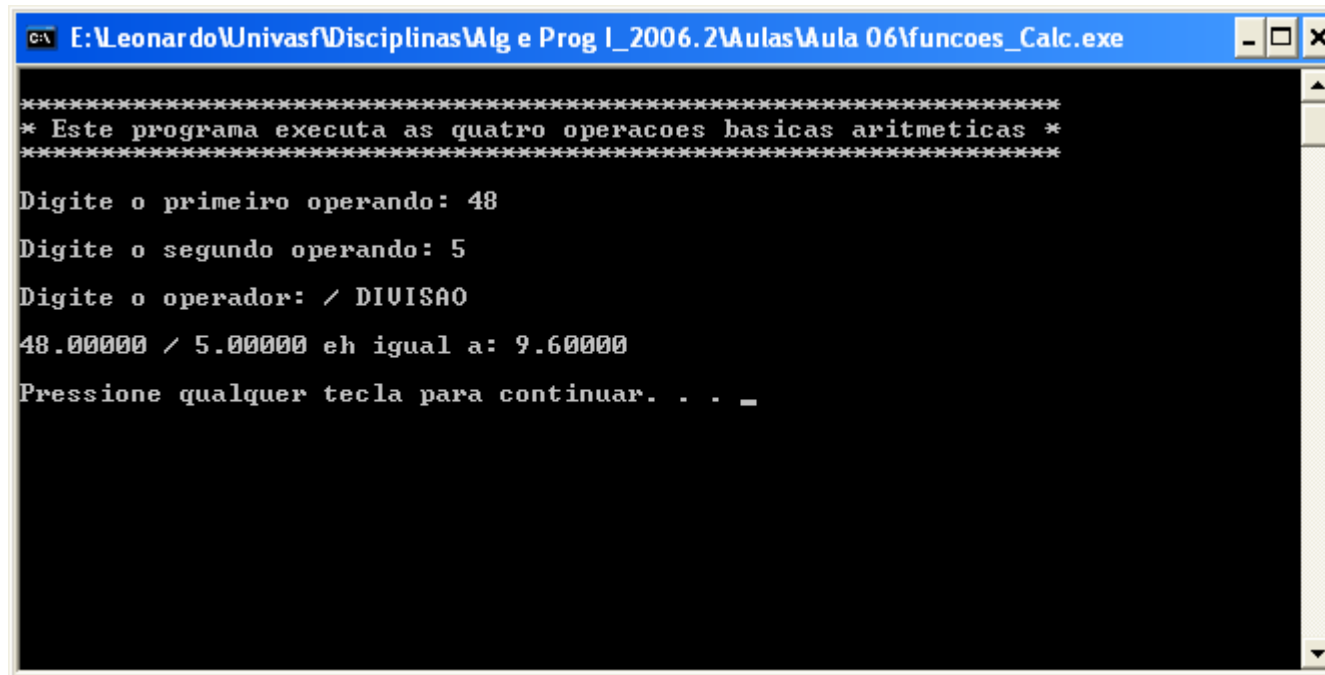
```
}
```

Protótipo da função calculadora

Chamada à função calculadora

Exemplo

- Uma saída sugerida para o programa anterior é:



```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_Calc.exe

*****
* Este programa executa as quatro operacoes basicas aritmeticas *
*****

Digite o primeiro operando: 48
Digite o segundo operando: 5
Digite o operador: / DIVISAO
48.00000 / 5.00000 eh igual a: 9.60000
Pressione qualquer tecla para continuar. . . _
```

Exemplo

- A função calculadora do programa anterior é definida da seguinte forma:

```
void calculadora(float a, float b, char op)
{
    switch (op)
    {
        case '+':
            printf(" SOMA\n\n%5.1f + %5.1f eh igual a: %5.1f\n\n", a, b, a+b);
            break;
        case '-':
            printf(" SUBTRACAO\n\n%6.1f - %6.1f eh igual a: %4.1f\n\n", a, b, a-b);
            break;
        case '*':
            printf(" MULTIPLICACAO\n\n%.2f * %.2f eh igual a: %.2f\n\n", a, b, a*b);
            break;
        case '/':
            printf(" DIVISAO\n\n%.5f / %.5f eh igual a: %.5f\n\n", a, b, a/b);
            break;
        default:
            printf(" %c\n\nOperacao desconhecida\n\n", op);
    }
}
```

Regras de escopo de funções

- A parte do programa na qual um identificador tem significado é conhecido como seu escopo;
- As variáveis variam seu escopo em três tipos:
 - Variáveis locais;
 - Parâmetros formais;
 - Variáveis globais.

Variáveis Locais

- São aquelas que só têm validade dentro do bloco no qual são declaradas;

Válida apenas no bloco definido dentro da função main()

```
#include<stdio.h>
/* Este programa demonstra o escopo de variáveis locais */
void funcao01();
void funcao02();
int main()
{
    int a=5;
    printf("Este programa demonstra o escopo de variáveis locais");
    printf("\n\nO Valor de a dentro da funcao main() eh: %d", a);
    funcao01();
    funcao02();
    {
        /* Bloco qualquer dentro da funcao main() */
        int a=20;
        printf("\n\nO valor de a dentro do bloco eh: %d", a);
    }
    system("pause");
    return 0;
}
void funcao01()
{
    int a=10;
    printf("\n\nO Valor de a dentro da funcao funcao01() eh: %d", a);
}
void funcao02()
{
    int a=15;
    printf("\n\nO Valor de a dentro da funcao funcao02() eh: %d", a);
}
```

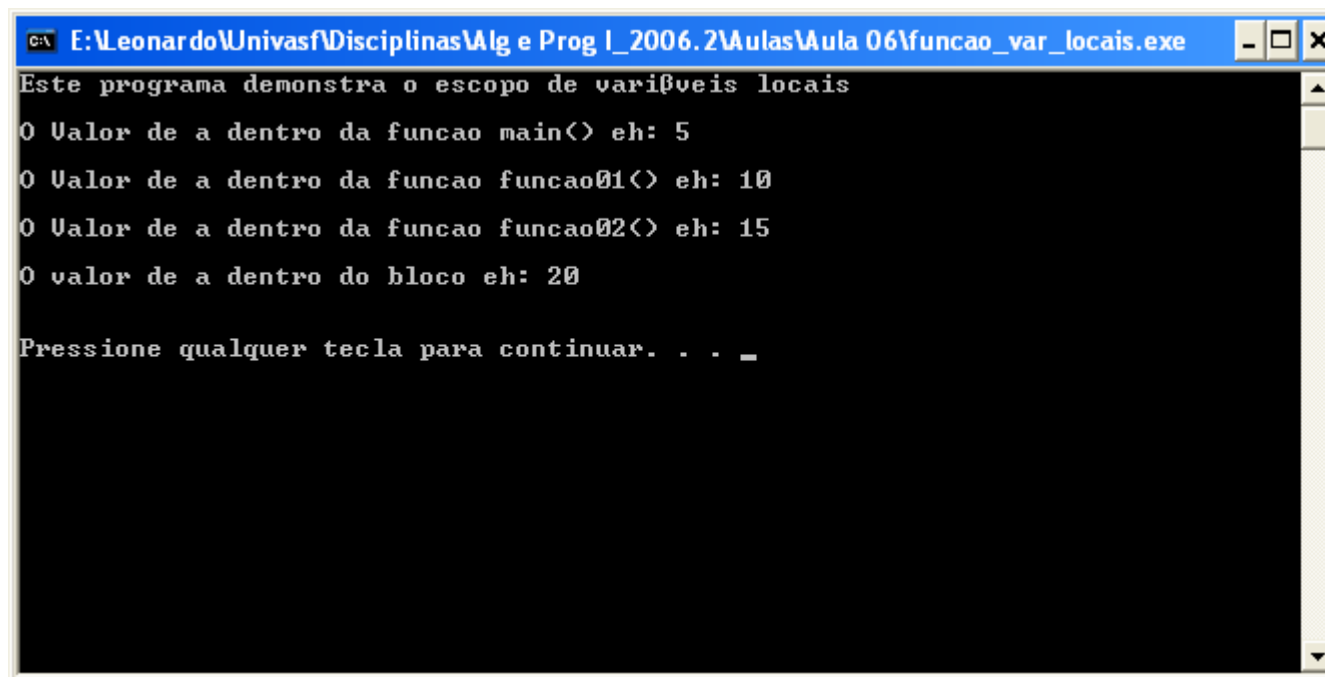
Válida apenas no bloco da função main()

Válida apenas no bloco da função funcao01()

Válida apenas no bloco da função funcao02()

Variáveis Locais

- A saída no console para o programa anterior será:



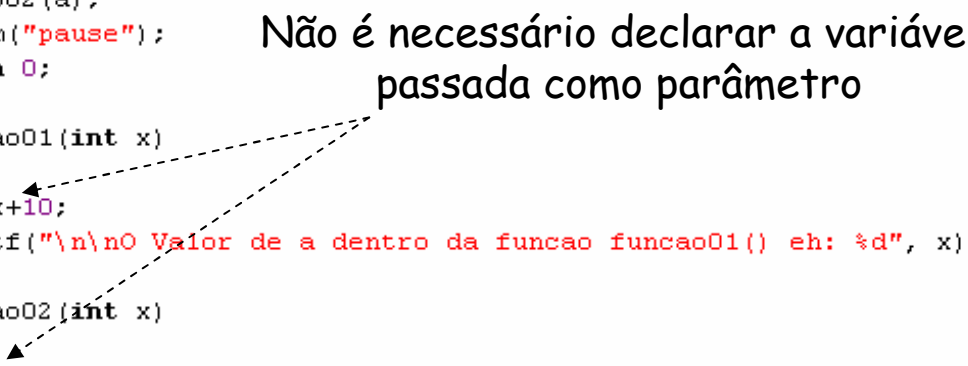
```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcao_var_locais.exe
Este programa demonstra o escopo de variáveis locais
O Valor de a dentro da funcao main() eh: 5
O Valor de a dentro da funcao funcao01() eh: 10
O Valor de a dentro da funcao funcao02() eh: 15
O valor de a dentro do bloco eh: 20
Pressione qualquer tecla para continuar. . . _
```

Parâmetros formais

- Estes são declarados como sendo as entradas de uma função. O parâmetro formal é uma variável local da função:

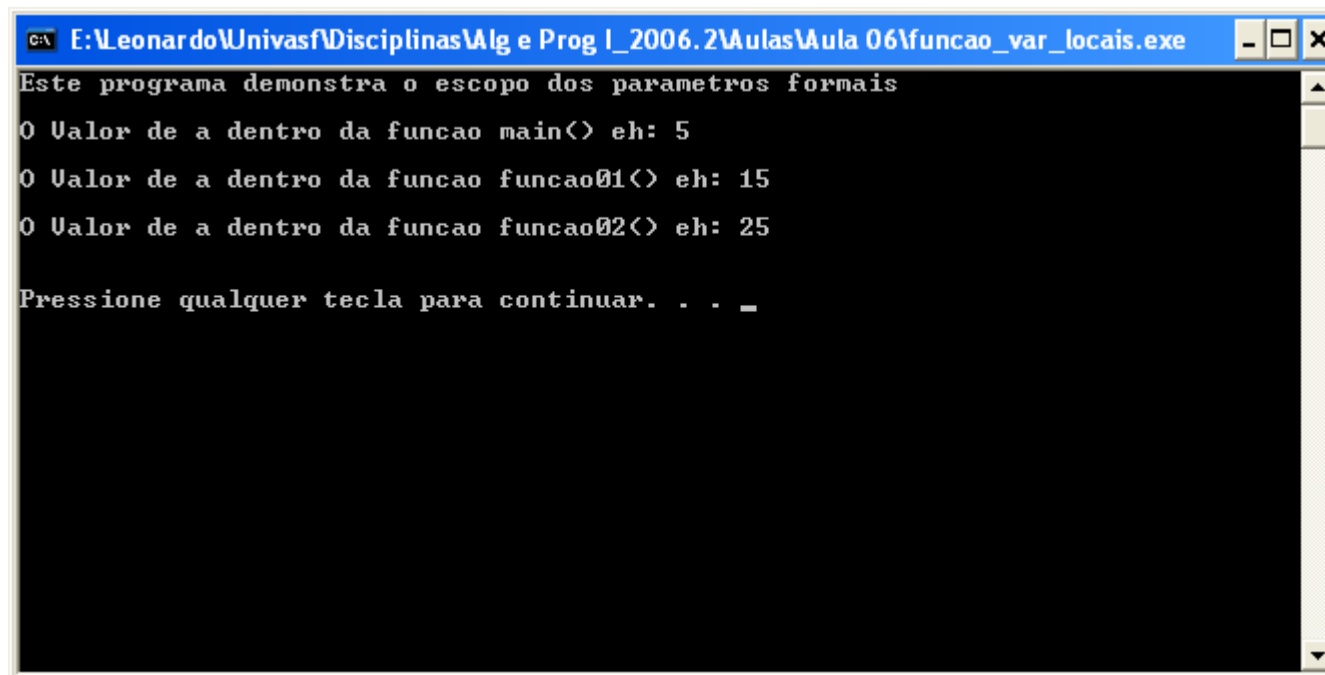
```
#include<stdio.h>
/* Este programa demonstra o escopo das variaveis globais */
void funcao01(int x);
void funcao02(int x);
int main()
{
    int a=5;
    printf("Este programa demonstra o escopo das variaveis globais");
    printf("\n\nO Valor de a dentro da funcao main() eh: %d", a);
    funcao01(a);
    funcao02(a);
    system("pause");
    return 0;
}
void funcao01(int x)
{
    x = x+10;
    printf("\n\nO Valor de a dentro da funcao funcao01() eh: %d", x);
}
void funcao02(int x)
{
    x = x + 20;
    printf("\n\nO Valor de a dentro da funcao funcao02() eh: %d\n\n\n", x);
}
```

Não é necessário declarar a variável passada como parâmetro



Parâmetros formais

- A saída no console para o programa anterior será:



```

E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcao_var_locais.exe
Este programa demonstra o escopo dos parametros formais
O Valor de a dentro da funcao main() eh: 5
O Valor de a dentro da funcao funcao01() eh: 15
O Valor de a dentro da funcao funcao02() eh: 25
Pressione qualquer tecla para continuar. . . _

```

Variáveis globais

- Variáveis são declaradas fora de todas as funções do programa;

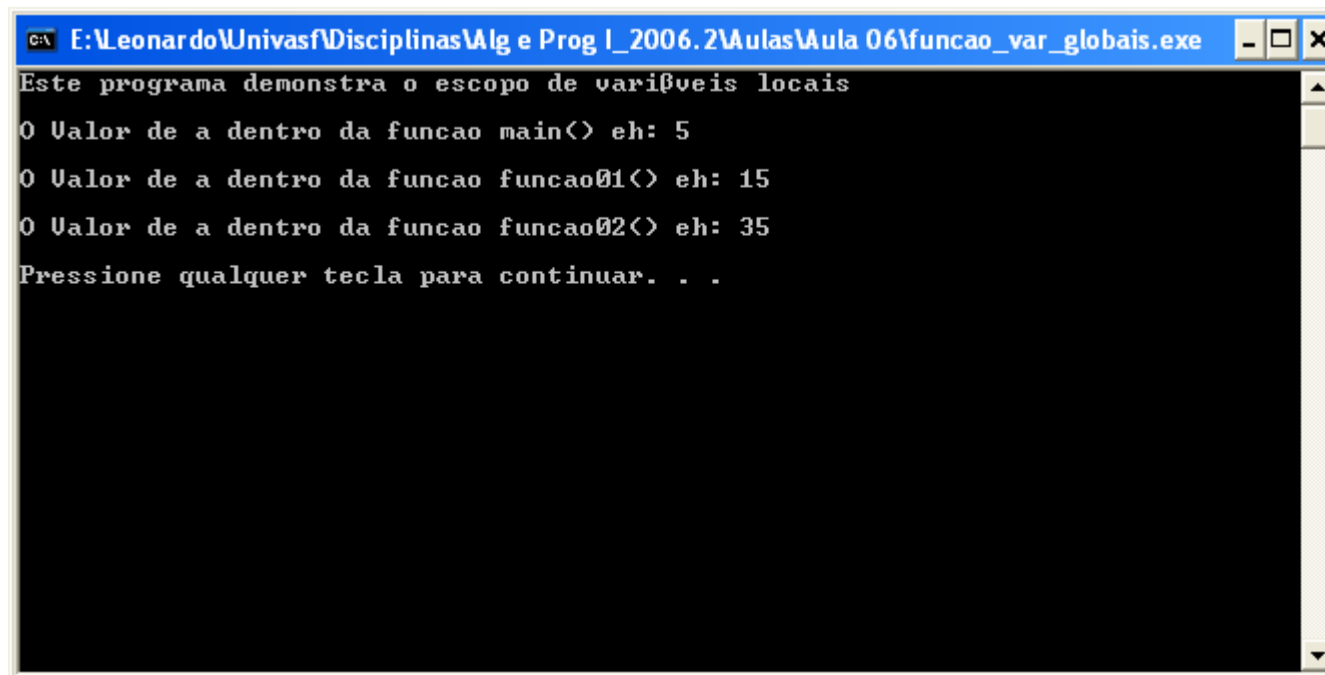
```
#include<stdio.h>
/* Este programa demonstra o escopo de variáveis locais */
int x=5;
void funcao01();
void funcao02();
int main()
{
    printf("Este programa demonstra o escopo de variáveis locais");
    printf("\n\nO Valor de a dentro da funcao main() eh: %d", x);
    funcao01();
    funcao02();
    system("pause");
    return 0;
}
void funcao01()
{
    x = x + 10;
    printf("\n\nO Valor de a dentro da funcao funcao01() eh: %d", x);
}
void funcao02()
{
    x = x + 20;
    printf("\n\nO Valor de a dentro da funcao funcao02() eh: %d\n\n", x);
}
```

Declaração da variável independente de função

Manipulação da mesma variável x

Variáveis globais

- A saída no console para o programa anterior será:



```

E:\LeonardoUnivasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcao_var_globais.exe
Este programa demonstra o escopo de variáveis locais
O Valor de a dentro da funcao main() eh: 5
O Valor de a dentro da funcao funcao01() eh: 15
O Valor de a dentro da funcao funcao02() eh: 35
Pressione qualquer tecla para continuar. . .

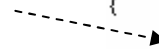
```

Variáveis globais

- Quando uma função tem uma variável local com o mesmo nome de uma variável global a função dará preferência à variável local;

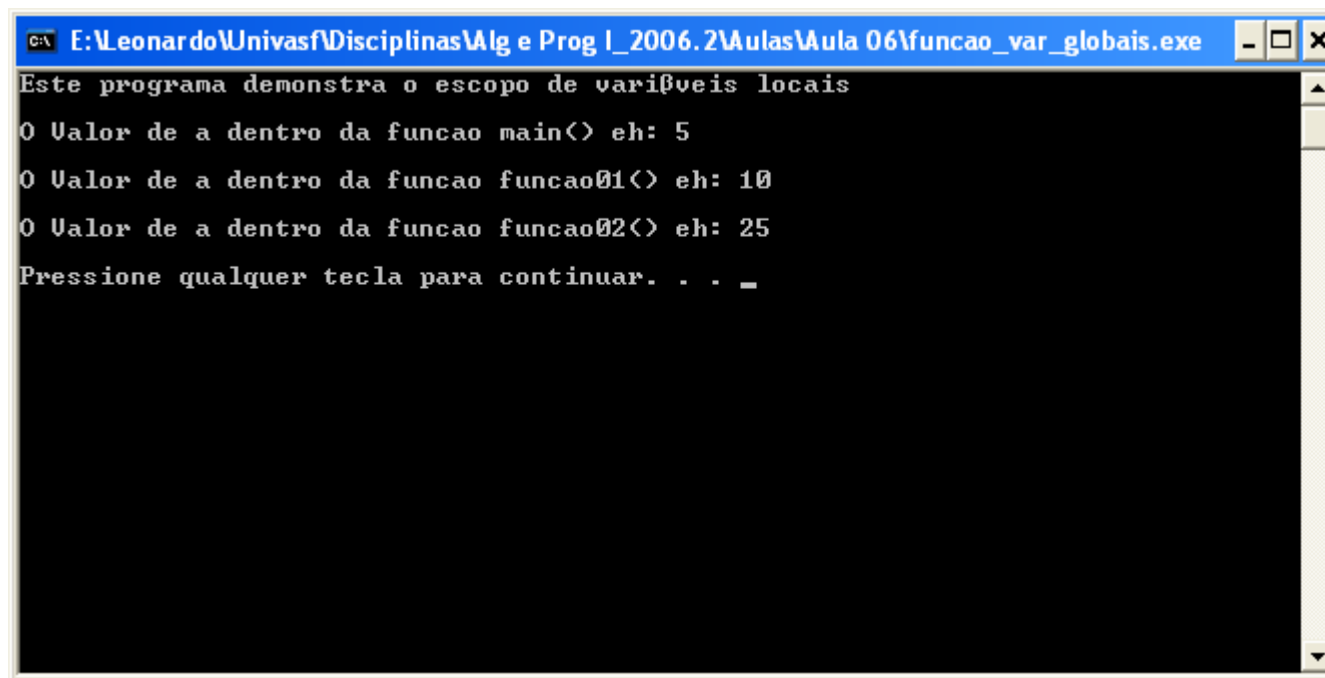
```
#include<stdio.h>
/* Este programa demonstra o escopo de variáveis locais */
int x=5;
void funcao01();
void funcao02();
int main()
{
    printf("Este programa demonstra o escopo de variáveis locais");
    printf("\n\nO Valor de a dentro da funcao main() eh: %d", x);
    funcao01();
    funcao02();
    system("pause");
    return 0;
}
void funcao01()
{
    int x = 10;
    printf("\n\nO Valor de a dentro da funcao funcao01() eh: %d", x);
}
void funcao02()
{
    x = x + 20;
    printf("\n\nO Valor de a dentro da funcao funcao02() eh: %d\n\n", x);
}
```

Variável local com o mesmo nome de uma variável global



Variáveis globais

- A saída no console para o programa anterior será:



```
C:\ E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcao_var_globais.exe
Este programa demonstra o escopo de variáveis locais
O Valor de a dentro da funcao main() eh: 5
O Valor de a dentro da funcao funcao01() eh: 10
O Valor de a dentro da funcao funcao02() eh: 25
Pressione qualquer tecla para continuar. . . _
```


Parâmetros de funções

- Vimos que os parâmetros de uma função é uma lista (0 ou mais) de tipos nomeados (variáveis);

tipo_de_retorno nome_da_funcao (parametros)

{

 corpo_da_funcao;

}

↑
Lista de "variáveis" (tipo nome1,
tipo nome2,..., tipo nomeX)

- Vejamos agora alguns casos particulares de parâmetros:


- Matrizes;
- Parâmetros da função main();

Parâmetros de funções

- Na declaração de uma função (protótipo) o nome das variáveis é opcional, porém, o tipo dessa variável é obrigatório;
- Na declaração de uma função (protótipo) a dimensão das matrizes é opcional mas o nome é obrigatório;

```
void localizaNum(int, int mat[]);
```

Nome opcional



The diagram illustrates the optional nature of parameter names and dimensions in a function prototype. It shows the code `void localizaNum(int, int mat[]);` with two dashed arrows pointing from labels below to specific parts of the code. One arrow points from 'Nome opcional' to the first parameter `int`, and another arrow points from 'Dimensão opcional' to the dimension part `[]` of the second parameter `mat[]`.

Dimensão opcional

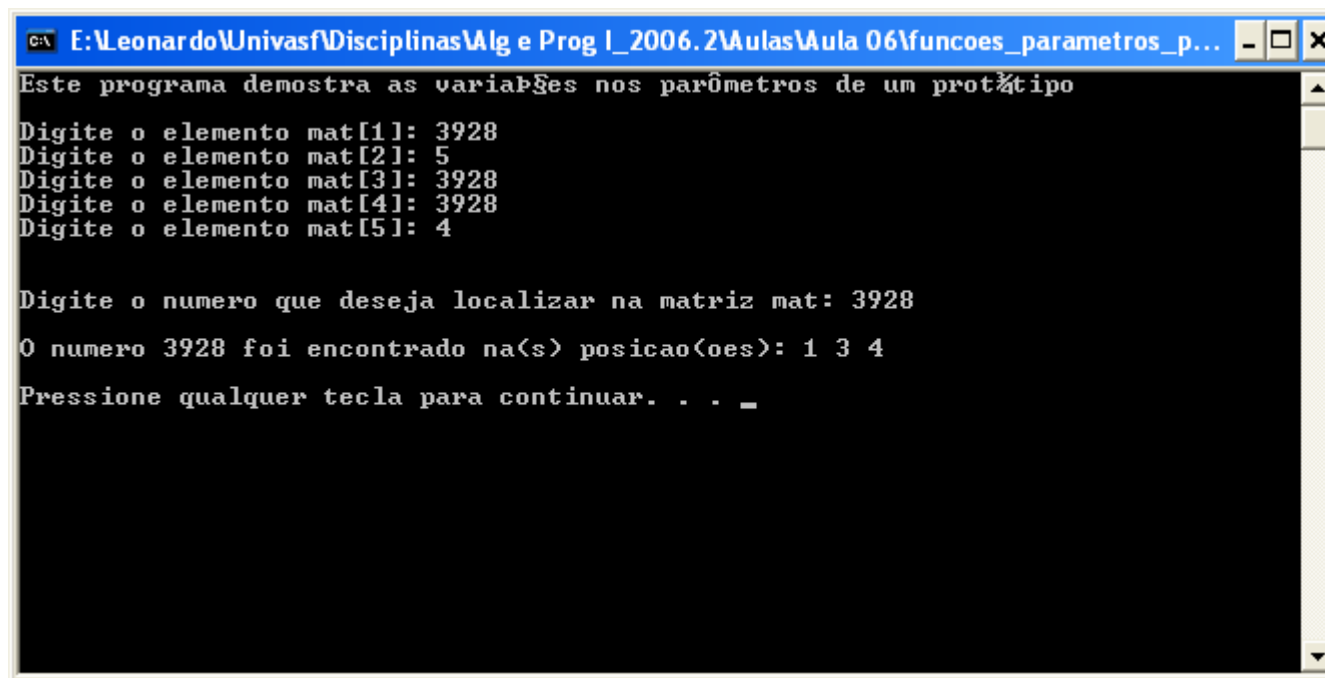
Parâmetros de funções

```
#include<stdio.h>
/* Este programa demonstra as variações nos parâmetros de um protótipo */
void localizaNum(int, int mat[]);
int main()
{
    int num, cont, mat[5];
    printf("Este programa demonstra as variações nos parâmetros de um protótipo\n\n");
    for(cont=0; cont<=4; cont++)
    {
        printf("Digite o elemento mat[%d]: ", cont+1);
        scanf("%d", &mat[cont]);
    }
    printf("\n\nDigite o numero que deseja localizar na matriz mat: ");
    scanf("%d", &num);
    localizaNum(num, mat);
    system("pause");
    return 0;
}
void localizaNum(int num, int mat[5])
{
    int i;
    printf("\nO numero %d foi encontrado na(s) posicao(oes): ", num);
    for(i=0; i<=4; i++)
    {
        if(mat[i]==num)
            printf("%d ", i+1);
    }
    printf("\n\n");
}
```

Parâmetros apenas com as definições básicas... Que o compilador necessitará

Parâmetros de funções

- A saída no console para o programa anterior será:



```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_parametros_p...
Este programa demonstra as variáveis nos parâmetros de um protótipo
Digite o elemento mat[1]: 3928
Digite o elemento mat[2]: 5
Digite o elemento mat[3]: 3928
Digite o elemento mat[4]: 3928
Digite o elemento mat[5]: 4

Digite o numero que deseja localizar na matriz mat: 3928
O numero 3928 foi encontrado na(s) posicao(oes): 1 3 4
Pressione qualquer tecla para continuar. . . _
```

Passagem de parâmetros

- Em geral, podem ser passados argumentos para sub-rotinas de duas maneiras;
 - Por valor e
 - Por referência;
- A **passagem de parâmetros por valor** copia o valor de um argumento no parâmetro formal da sub-rotina;
- Alterações feitas nos parâmetros da sub-rotina **não têm nenhum efeito nas variáveis** usadas para chamá-la;


Passagem por valor

- Vejamos um exemplo com passagem de parâmetros por valor;

```
#include<stdio.h>
/* Este programa demonstra a passagem de parâmetros por valor */
void randomica(int x);
int main()
{
    int x;
    printf("Este programa demonstra a passagem de parâmetros por valor");
    printf("\n\nDigite um valor para x: ");
    scanf("%d", &x);
    randomica(x);
    printf("\nO valor de x apos a funcao randomica eh: %d\n\n", x);
    system("pause");
    return 0;
}

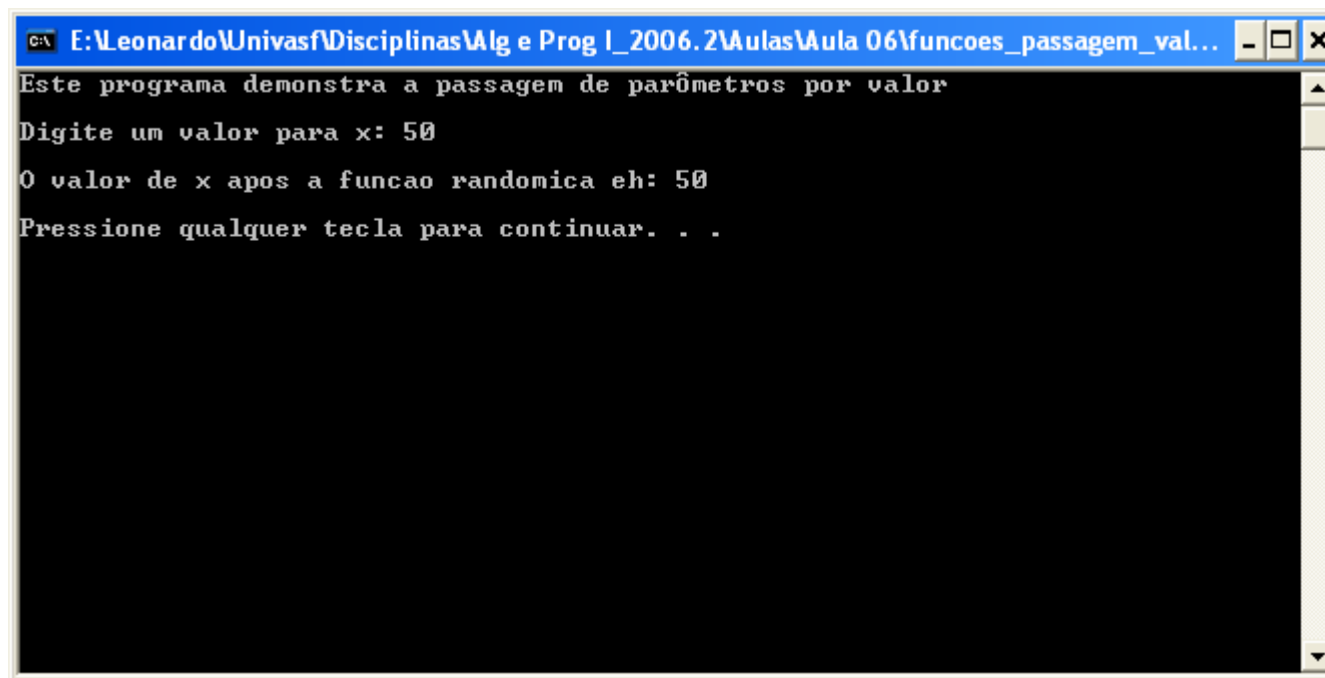
void randomica(int x)
{
    x = rand();
}
```

Função que retorna um inteiro aleatório menor que 32767



Passagem por valor

- A saída no console para o programa anterior será:



A screenshot of a Windows command prompt window. The title bar shows the file path: `E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_passagem_val...`. The window has standard minimize, maximize, and close buttons. The text inside the window is as follows:

```
Este programa demonstra a passagem de parâmetros por valor
Digite um valor para x: 50
O valor de x apos a funcao randomica eh: 50
Pressione qualquer tecla para continuar. . .
```

Passagem por referência

- Outro tipo de passagem de parâmetros para uma função é a **passagem de parâmetros por referência**;
- Neste caso as alterações nos parâmetros formais, dentro da função, alteram os valores dos parâmetros que foram passados para a função;
- A idéia aqui é passar a **referência ao endereço de memória** da variável (Ponteiro) e não mais o valor da variável para a função;

Passagem por referência

- A única no código-fonte é **acrescentar um &** na frente das variáveis que estivermos passando para a função;
- A "alteração" na sintaxe das funções é o **acréscimo do *** na frente das variáveis referenciadas;
- Os operadores unários & e * serão detalhados na aula de ponteiros;

Passagem por referência

- Vejamos o exemplo anterior com passagem de parâmetros por referência;

```
#include<stdio.h>
/* Este programa demonstra a passagem de parâmetros por referência */
void randomica(int *x);
int main()
{
    int x;
    printf("Este programa demonstra a passagem de parâmetros por referencia");
    printf("\n\nDigite um valor para x: ");
    scanf("%d", &x);
    randomica(&x);
    printf("\nO valor de x apos a funcao randomica eh: %d\n\n", x);
    system("pause");
    return 0;
}

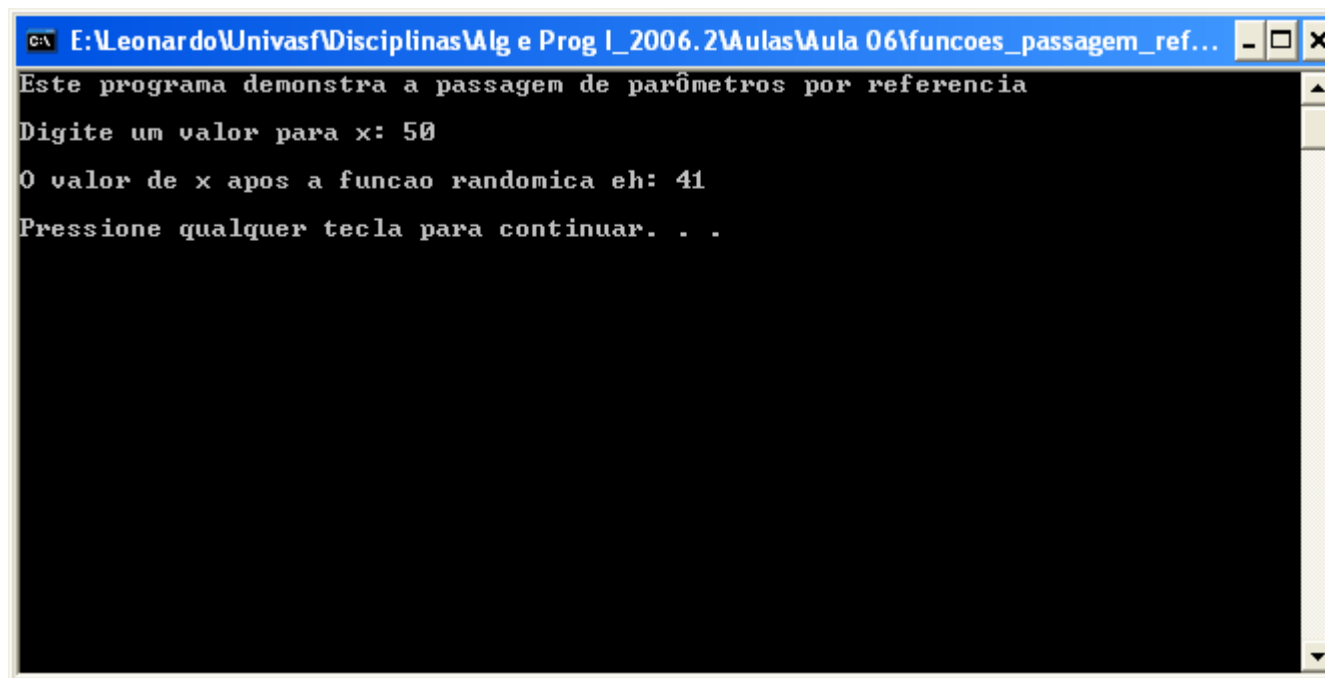
void randomica(int *x)
{
    *x = rand();
}
```

Utilização do operador * na função

Utilização do operador & na passagem do parâmetro

Passagem por referência

- A saída no console para o programa anterior será:



The screenshot shows a Windows command prompt window with the title bar "E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 06\funcoes_passagem_ref...". The window contains the following text:

```
Este programa demonstra a passagem de parâmetros por referencia
Digite um valor para x: 50
O valor de x apos a funcao randomica eh: 41
Pressione qualquer tecla para continuar. . .
```

Bibliografia

- SCHILDT H. *"C Completo e Total"*, Makron Books. SP, 1997.
- MIZRAHI, V. V. *"Treinamento em Linguagem C++ Módulo 1"*, Makron Books, SP, 1995.
- UFMG *"Curso de Linguagem C"*, Universidade Federal de Minas Gerais.