

INF01202

Algoritmos e Programação

Modalidade Ead – Turma H

**Material de apoio: capacitar no uso de funções
com parâmetros e de protótipos**

Funções com parâmetros

Seja o programa **termosfibv1** que apresenta os 20 primeiros termos da série de Fibonacci

//calcular e apresentar os 20 primeiros termos da serie de fibonacci

#include <stdio.h>

#include <stdlib.h>

void termos_fibo_fixos (void)

{

int i, fibo1, fibo2, fibotot;

fibo1 = 1;

fibo2 = 1;

fibotot = 1;

printf("\n%8d \n%8d", fibo1, fibo2);

for (i = 3; i <= 20; i++)

{

fibo1 = fibo2;

fibo2 = fibotot;

fibotot = fibo1 + fibo2;

printf("\n%8d", fibotot);

}

}

int main ()

{

printf("\nOs 20 primeiros termos da serie de Fibonacci:\n");

termos_fibo_fixos();

printf("\n");

system("pause");

return 0;

}

declaração da função
termos_fibo_fixos

chamada da função
termos_fibo_fixos

Execução do programa termosfibv1

 E:\EAD2007\FUNCOES\termosfibv1.exe

Os 20 primeiros termos da serie de Fibonacci:

1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765

Pressione qualquer tecla para continuar. . .

O programa **termosfibv1** sempre que executar vai apresentar os mesmos termos da série de Fibonacci, pois o número de termos é a constante 20.

Para que o número de termos se altere, o código do programa deve ser alterado.

Mas se o desejado for, a cada execução escolher-se, um número diverso de termos da série a serem apresentados, sem se precisar alterar o código??

Como fazê-lo?

Resposta: no processo de ativação da função, passar a ela a informação do número de termos que se deseja que sejam apresentados, ou seja, passar esse número como um **parâmetro para a função.**

PARÂMETROS

- Eles são declarados no cabeçalho das funções e são locais a elas. Veja no exemplo:

```
.....  
#include <stdlib.h>  
void termos_fibo_variaveis (int n)  
{  
    .....
```

n é um
parâmetro e é
declarado com
seu tipo (int)

PARÂMETROS

A comunicação das funções com o meio externo, ou seja, com os programas que as chamam, deve ocorrer basicamente através de parâmetros. Veja:

.....

```
void termos_fibo_variaveis (int n)
```

```
{
```

```
    int i, fibo1, fibo2, fibotot;
```

```
    if (n == 1)
```

```
        printf("\n      1");
```

```
    else .....
```

```
        for (i = 3; i <= n; i++)
```

```
    ...
```

```
}
```

```
....
```

```
int main ( )
```

```
{
```

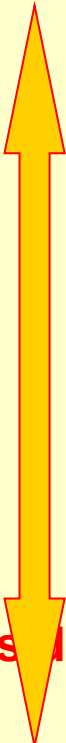
```
    .....printf("\nQuantos termos da serie deseja? ");
```

```
    .....
```

```
    termos_fibo_variaveis(num_ter);
```

```
    .....    return 0;
```

```
}
```



A comunicação é feita pela variável **num_ter** do main através do parâmetro **n** da função termos_fibo_variaveis

PARÂMETROS

Toda função que apresentar parâmetros na sua declaração, deve ser chamada com argumentos (parâmetros) de igual número e tipo, acompanhando na chamada a ordem de sua declaração. Veja:

```
.....  
void divisao(float num1, float num2) // declaração de divisao  
{  
    printf("\n%6.2f dividido por %6.2f = %6.2f", num1 , num2 , num1 /  
    num2 );  
}
```

```
.....  
int main ( )  
{  
    .....  
    divisao(numero1, numero2); // chamada  
    .....  
    return 0;  
}
```

Chamada da função
divisao com 2
parâmetros float, mesmo
número e tipo da
declaração;
numero1 corresponde a
num1,
numero 2 a num2

PARÂMETROS (resumo)

ATENÇÃO:

Os nomes das variáveis declaradas no cabeçalho de uma função são independentes dos nomes das variáveis usadas para chamar a mesma função.

As declarações de **main** são usadas em **main**; as declarações de uma particular função são usadas somente dentro dela.

E, os parâmetros declarados no cabeçalho de uma função também somente existem dentro da função onde estão declarados.

Ex.:

no programa **termosfibv2** (a seguir), na declaração da função **termos_fibo_variaveis** foi declarado **int n** (declarado no cabeçalho da função), na chamada foi usado **num_ter** (de tipo inteiro), declarado em **main**, onde está sendo usado.

Seja o programa **termosfibv2** que apresenta os **n** primeiros termos da série de Fibonacci

```
//calcula e apresenta os n primeiros termos da serie de fibonacci
```

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

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

```
void termos_fibo_variaveis (int n)
```

```
{  
    int i, fibo1, fibo2, fibotot;  
    if (n == 1)  
        printf("\n      1");  
    else  
        if (n == 2)  
            printf("\n      1 \n      1");  
        else  
        {  
            fibo1 = 1;  
            fibo2 = 1;  
            fibotot = 1;  
            printf("\n%8d \n%8d", fibo1, fibo2);  
            for (i = 3; i <= n; i++)  
            {  
                fibo1 = fibo2;  
                fibo2 = fibotot;  
                fibotot = fibo1 + fibo2;  
                printf("\n%8d", fibotot);  
            }  
        }  
}
```

Declaração de **n**
em **termos_fibo_variaveis**

declaração da função
termos_fibo_variaveis,
com um parâmetro de
tipo inteiro **n**.

cont.

Seja o programa **termosfibv2** que apresenta os **n** primeiros termos da série de Fibonacci

```
int main ( )
{
    system("color 70");
    int num_ter;
    do
    {
        printf("\nQuantos termos da serie deseja? ");
        scanf("%d", &num_ter);
        if (num_ter < 1)
            printf("\nNumero de termos deve ser >= 1\n");
    }
    while (num_ter < 1);
    printf("\nO(s) %d primeiro(s) termo(s) da serie de Fibonacci:\n", num_ter);
    termos_fibo_variaveis(num_ter);
    printf("\n");
    system("pause");
    return 0;
}
```

Declaração de **num_ter**
em **main**

chamada da função
termos_fibo_variaveis, com um
argumento de tipo inteiro,
num_ter

Execuções do programa termosfibv2

 G:\EAD2007\FUNCOES\termosfibv2.exe

Quantos termos da serie deseja? 3

O(s) 3 primeiro(s) termo(s) da serie de Fibonacci:

1
1
2

Pressione qualquer tecla para continuar. . . _

 G:\EAD2007\FUNCOES\termosfibv2.exe

Quantos termos da serie deseja? 8

O(s) 8 primeiro(s) termo(s) da serie de Fibonacci:

1
1
2
3
5
8
13
21

Pressione qualquer tecla para continuar. . .

PASSAGEM DE PARÂMETROS(resumo)

Ao ser ativada, após chamada, a função **termos_fibo_variaveis**, a variável **n** é criada.

E o valor existente nesse momento em **num_ter** é transferido para **n**.

A conexão entre **num_ter** e **n** só existe no momento que a função é ativada.

Qualquer modificação de **n** que acontecer a partir deste ponto só é conhecida e percebida dentro da função **termos_fibo_variaveis**.

```
void termos_fibo_variaveis (int n)
```



Atenção:
n existe na função **termos_fibo_variaveis**.
num_ter existe in **main**.

Chamada na função **main**:

```
termos_fibo_variaveis(num_ter);
```

Seja o programa **divisao** que realiza a divisão de dois números reais

```
//apresenta o resultado da divisão de dois números
#include <stdio.h>
#include <stdlib.h>

void divisao(float num1, float num2)
{
    printf("\n%6.2f dividido por %6.2f = %6.2f", num1 , num2 , num1 / num2 );
}

int main ( )
{
    system("color 70");
    float numero1, numero2;
    printf("\nPara a divisao, forneça o primeiro valor: ");
    scanf("%f", &numero1);

    do
    {
        printf("\nForneca agora o segundo valor (diferente de 0): ");
        scanf("%f", &numero2);
        if (numero2 == 0)
            printf("\nEsse valor deve ser diferente de 0!\n");
    }
    while (numero2 == 0);
    divisao(numero1, numero2);
    printf("\n");
    system("pause");
    return 0;
}
```

declaração da função **divisao**, com dois parâmetros de tipo float, **num1** e **num2**.

chamada da função divisao, com dois argumentos de tipo float, **num1** e **num2**.

Execução do programa **divisao** que realiza a divisão de dois números reais

 G:\EAD2007\FUNCOES\funccomdoisparam.exe

Para a divisao, forneça o primeiro valor: 45

Forneça agora o segundo valor (diferente de 0): -1.5

45.00 dividido por -1.50 = -30.00

Pressione qualquer tecla para continuar. . .

PASSAGEM DE PARÂMETROS

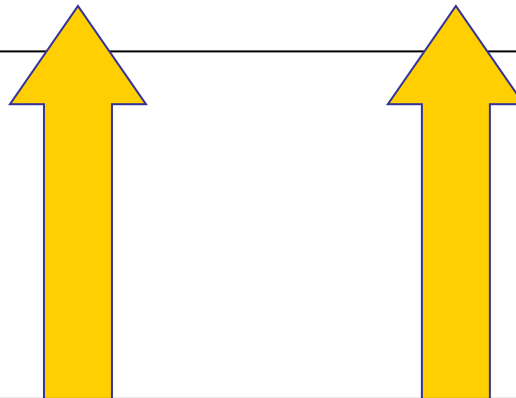
Ao ser ativada a função **divisao**, **num1** e **num2** são criados.

E os valores existentes nesse momento em **numero1** e **numero2** são copiados respectivamente para **num1** e **num2**.

A conexão entre **numero1** e **num1** e **numero2** e **num2** só existe no momento de ativação da função.

Quaisquer modificações de **num1** e **num2** que acontecerem a partir desse ponto só serão conhecidas e percebidas dentro da função **divisao**.

```
void divisao(float num1, float num2)
```



Atenção:

num1 e **num2** existem na função **divisao**.

numero1 e **numero2** existem in **main**.

Chamada na função **main**:

```
divisao(numero1, numero2);
```


Seja o programa **alteratexto** que escreve um texto repetindo n vezes uma letra informada

```
//escreve um texto repetindo n vezes uma letra informada
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

void variasletras(int vezes , char carac)
{

    char texto[31];
    int i, j;
    fflush(stdin);
    printf("\nForneca um texto com no maximo 30 caracteres: ");
    gets(texto);
    for (i = 0; i<=strlen(texto); i++)
    {
        if (texto[i] == carac || texto[i] == toupper(carac))
        {
            for (j = 1; j <vezes; j++)
                printf("%c", texto[i]);
            printf("%c", texto[i]);
        }
    }
}
```

declaração da função **variasletras**, com dois parâmetros um de tipo inteiro, **vezes** e outro de tipo char, **carac**.

...

Seja o programa **alteratexto** que escreve um texto repetindo n vezes uma letra informada

```
int main ( )
{
    system("color 70");
    int vezes;
    char letra;
    do
    {
        printf("\nQual a letra a repetir? ");
        scanf("%c", &letra);
        letra = tolower(letra);
        if (letra < 'a' || letra > 'z')
            printf("\nDeve ser fornecida uma letra!");
    }
    while (letra < 'a' || letra > 'z');
    do
    {
        printf("\nQuantas vezes letra deve ser repetida? (maior que 1!): ");
        scanf("%d", &vezes);
        if (vezes < 2)
            printf("\nNumero de vezes nao deve ser inferior a 2!\n");
    }
    while (vezes < 2);
    variasletras(vezes, letra);
    printf("\n");
    system("pause");
    return 0;
}
```

chamada da função
variasletras, com dois
parâmetros um de tipo inteiro,
vezes, e outro de tipo char,
carac.

Execução do programa **alteratexto** que escreve um texto repetindo n vezes uma letra informada

 G:\EAD2007\FUNCOES\funcocomumparaminteumchar.exe

Qual a letra a repetir? i

Quantas vezes a letra deve ser repetida? (maior que 1!): 4

Forneça um texto com no maximo 30 caracteres: RIO de janeiro

RIIIIO de janeiiiiiro

Pressione qualquer tecla para continuar. . .

Em todos os programas recém examinados:

termosfibv1,
termosfibv2,
divisao,
alteratexto

as declarações das funções estão sempre antes da função **main**.

Isso porque em C, antes de usar-se alguma variável, função, etc. deve-se primeiro declará-la.

A seguir, uma tentativa de usar uma função sem antes declará-la e os erros acusados.

Para tanto o programa **divisao** foi alterado. Observem a seguir:

Seja o programa **divisaoerr**,
que realiza a divisão de dois números reais,
com declaração de função após a tentativa de usá-la.

```
#include <stdio.h>
#include <stdlib.h>
int main ( )
{
    system("color 70");
    float numero1, numero2;
    printf("\nPara a divisao, forneça o primeiro valor: ");
    scanf("%f", &numero1);

    do
    {
        printf("\nForneça agora o segundo valor (diferente de 0): ");
        scanf("%f", &numero2);
        if (numero2 == 0)
            printf("\nEsse valor deve ser diferente de 0!\n");
    }
    while (numero2 == 0);
    divisao(numero1, numero2);
    printf("\n");
    system("pause");
    return 0;
}

void divisao(float num1, float num2)
{
    printf("\n%6.2f dividido por %6.2f = %6.2f", num1 , num2 , num1 / num2 );
}
```

chamada da função
divisao, antes de sua
declaração

declaração da função
divisao, com dois
parâmetros de tipo
float (**num1** e **num2**).

Execução do programa `divisaoerr`
que realiza a divisão de dois números reais,
com declaração de função após a tentativa de usá-la.

20	G:\EAD2007\FUNCOES\funccomdoi...	'divisao' undeclared (first use this function) (Each undeclared identifier is reported only once for each function it appears in.)
	G:\EAD2007\FUNCOES\funccomdoi...	In function 'void divisao(float, float)':
26	G:\EAD2007\FUNCOES\funccomdoi...	'void divisao(float, float)' used prior to declaration

Os erros de compilação acusam que há uma função não declarada e logo em seguida que a função foi usada antes de sua declaração.

Protótipos de funções

Protótipos de funções:

Para contornar a exigência de primeiro ter-se que declarar uma função para só depois poder-se usá-la, bem como para permitir que o sistema tenha um controle mais completo sobre os parâmetros que devem ser utilizados com as funções, a ponto de poder acusar erros se os parâmetros forem fornecidos de forma incorreta, foram criados os **protótipos de funções**.

Um **protótipo de função** compreende o tipo da função, seu nome e sua lista de parâmetros. É tipicamente é colocado logo no início do programa antes da função **main**.

Após a função **main**, as funções são declaradas de modo completo.

Protótipos das funções para as utilizações nos programas anteriores:

```
void termos_fibo_fixos (void);  
void termos_fibo_variaveis (int n);  
void divisao(float num1, float num2);  
void variasletras(int vezes , char carac);
```


Protótipos de funções (cont.)

Particularidade na declaração dos protótipos das funções:

ao indicar os parâmetros, é desnecessário mencionar seus nomes. O sistema só precisa saber seus tipos, na ordem em que serão utilizados.

Protótipos das funções utilizadas nos programas anteriores, só indicando o tipo dos parâmetros:

```
void termos_fibo_fixos (void);  
void termos_fibo_variaveis (int);  
void divisao(float, float);  
void variasletras(int, char);
```

Seja o programa termosfibv1comprot, versão do programa termosfibv1 usando protótipo

//calcular e apresentar os 20 primeiros termos da serie de fibonacci

```
#include <stdio.h>
#include <stdlib.h>
void termos_fibo_fixos (void);
int main ( )
{
    printf("\nOs 20 primeiros termos da serie de Fibonacci:\n");
    termos_fibo_fixos( );
    printf("\n");
    system("pause");
    return 0;
}
```

protótipo da função
termos_fibo_fixos

chamada da função
termos_fibo_fixos

```
void termos_fibo_fixos (void)
{
    int i, fibo1, fibo2, fibotot;
    fibo1 = 1;
    fibo2 = 1;
    fibotot = 1;
    printf("\n%8d \n%8d", fibo1, fibo2);
    for (i = 3; i <= 20; i++)
    {
        fibo1 = fibo2;
        fibo2 = fibotot;
        fibotot = fibo1 + fibo2;
        printf("\n%8d", fibotot);
    }
}
```

declaração da função
termos_fibo_fixos

Cuidados na passagem de parâmetros

Seja o programa **alteratextoerr** que escreve um texto repetindo n vezes uma letra informada com chamada de função com parâmetros trocados

```
//escreve um texto repetindo n vezes uma letra informada
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
void variasletras(int , char);
int main ( )
{
    system("color 70");
    int vezes;
    char letra;
    do
    {
        printf("\nQual a letra a repetir? ");
        scanf("%c", &letra);
        letra = tolower(letra);
        if (letra < 'a' || letra > 'z')
            printf("\nDeve ser fornecida uma letra!");
    }
    while (letra < 'a' || letra > 'z');
    do
    {
        printf("\nQuantas vezes a letra deve ser repetida? (maior que 1!): ");
        scanf("%d", &vezes);
        if (vezes < 2)
            printf("\nNumero de vezes nao deve ser inferior a 2!\n");
    }
    while (vezes < 2);
    variasletras(letra, vezes);
    printf("\n");
    system("pause");
    return 0;
}
...
```

Tanto no protótipo da função **variasletras**, quanto em sua declaração (ver *slide seguinte*) os parâmetros da função indicados são de tipo: **int e char**, nesta ordem.

Mas na chamada de **variasletras** os parâmetros foram invertidos:

char e int, e não como na declaração da função, **int e char**.

Seja o programa **alteratextoerr**
que escreve um texto repetindo n vezes uma letra informada
com chamada de função com parâmetros trocados

```
void variasletras(int vezes , char carac)
{

    char texto[31];
    int i, j;
    fflush(stdin);
    printf("\nForneca um texto com no maximo 30 caracteres: ");
    gets(texto);
    for (i = 0; i<=strlen(texto); i++)
    {
        if (texto[i] == carac || texto[i] == toupper(carac))
        {
            for (j = 1; j <vezes; j++)
                printf("%c", texto[i]);
            printf("%c", texto[i]);
        }
    }
}
```

**Execução do programa `alteratextoerr`
que escreve um texto repetindo n vezes uma letra informada
com chamada de função com parâmetros trocados**

 G:\EAD2007\FUNCOES\alteratextoerr.exe

Qual a letra a repetir? i

Quantas vezes a letra deve ser repetida? (maior que 1!): 4

Forneça um texto com no maximo 30 caracteres: Rio de janeiro

Rio de janeiro

Pressione qualquer tecla para continuar. . .

Cuidados na passagem de parâmetros:

Os argumentos das chamadas devem corresponder em número e tipo aos parâmetros colocados na declaração da função.

Como pode-se observar no programa recém apresentado, `alteratextoerr`, versão alterada do programa `alteratexto`, na chamada da função `variasletras` os parâmetros foram invertidos.

O que aconteceu?

O programa executou, mas não fez o que era previsto.

Logo, o programador deve verificar atentamente a ordem e tipo dos parâmetros que está utilizando, uma vez que seus enganos nem sempre serão detectados pelo sistema.