

Funções

Aula 08

Marcos Silvano Almeida

marcossilvano@professores.utfpr.edu.br

Departamento de Computação

UTFPR Campo Mourão

BCC31A

Funções

- Uma função é um subprograma
 - Sequência de comandos para realizar uma tarefa específica
 - Pequenos módulos de programa que favorecem a reutilização de código
 - Escondem a complexidade da implementação: como funciona o printf?
- Sintaxe de uso da função (chamada):
 - `nome_da_função(int param1, int param2, ...)`

```
// função que imprime na saída padrão (terminal)
```

```
printf("Exemplo de saída: %d e %d\n", a, b);
```

```
// função que lê da entrada padrão (terminal)
```

```
scanf("%d %d", &a, &b);
```

```
// retorna um número aleatório entre 0 e 2.147.483.647 (RAND_MAX)
```

```
int sorteio = rand() % 10; // 0 a 9
```

As funções estão contidas nas bibliotecas

- As funções que utilizamos até o momento estão escritas em bibliotecas
 - As bibliotecas são escritas na linguagem C
 - Usam a linguagem para criar funcionalidades reutilizáveis
- As bibliotecas contém
 - Valores padrões (macros): INT_MAX, INT_MIN, RAND_MAX, ...
 - Funções: printf(), scanf(), rand(), sqrt(), ...
 - Tipos customizados e estruturados ([em outra aula](#))

```
printf("Resultado: %d e %d\n", a, b);           // biblioteca <stdio.h>

scanf("%d %d", &a, &b);                         // biblioteca <stdio.h>

int sorteio = rand() % 10; // 0 a 9              // biblioteca <stdlib.h>
```

Chamando funções: parâmetros e retorno

- Funções podem receber parâmetros para o seu funcionamento

```
printf("Resultado: %d e %d\n", a, b);
```

```
scanf("%d %d", &a, &b);
```

- Funções podem retornar um valor ao finalizarem

```
// printf retorna a quantidade de caracteres escritos na saída
```

```
int qte = printf("Resultado: %d e %d\n", a, b);
```

```
int sorteio = rand() % 10; // 0 a 9
```

- Quando uma função é chamada, o **código chamador** fica parado na linha da função e só volta a executar quando a **função finalizar**

Declarando funções

- Um programa C inicia pela função `main()`
 - Até agora, a única função que escrevemos foi a `main()`
- Sintaxe da declaração de uma função

```
tipo_retorno nome_da_função(int param1, int param2,...) {  
    ...  
    return valor; // se houver tipo de retorno  
}
```

- Observe a declaração da função `main()`

```
int main() {  
    // conteúdo...  
    return 0; // deve retornar int  
}
```

Return define os pontos de saída da função.

tipo_retorno = int, float, char...
Encerra a função e retorna valor
(return é obrigatório)

tipo_retorno = void
Encerra a função mais cedo
(opcional para void)

Declarando e chamando função max(a,b)

```
int max1(int a, int b) {  
    int maior;  
    if (a > b) {  
        maior = a;  
    } else {  
        maior = b;  
    }  
    return maior;  
}
```



```
int main() {  
    int x = 5, y = 10;  
    int res = max1(x, y);  
    printf("MAIOR: %d\n", res);  
    printf("MAIOR: %d\n", max2(x, y));  
    return 0;  
}
```

```
int max2(int a, int b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

As funções max() e max2() têm o mesmo propósito: retornar o maior entre os dois parâmetros passados.

Demonstram duas abordagens distintas de retorno dentro de uma função.

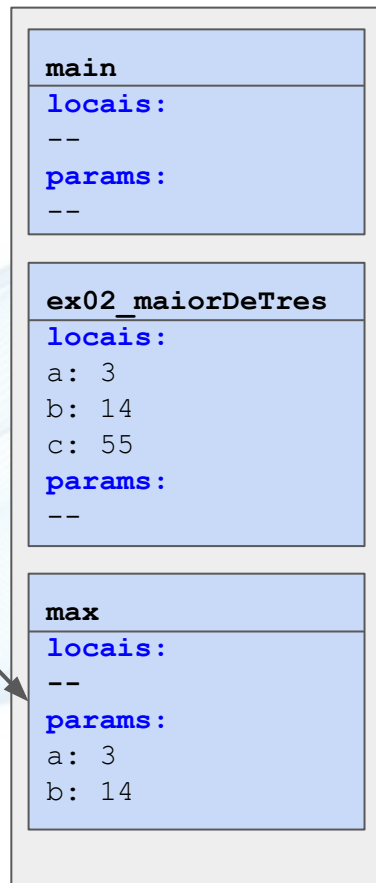
Vamos implementar outras funções

1. Média aritmética de 3 valores
2. Conceito (ABCDF) sobre nota
3. Imprimir tabela ASCII (33 ao 126)
4. Imprimir um número por extenso (1 dígito)
5. Imprimir um número por extenso (2 dígitos)
6. Dada a dimensão, imprimir caixa  com X
7. Dada a dimensão, Imprimir triângulo  com X

A pilha de chamadas de funções

```
int max(int a, int b) {  
    if (a > b) return a;  
    else      return b;  
}  
  
void ex02_maiorDeTres() {  
    int a, b, c;  
    printf("Informe 3 valores: ");  
    scanf("%d %d %d", &a, &b, &c);  
  
    printf("MAX:%d\n", max(max(a,b), c));  
}  
  
int main() {  
    ex02_maiorDeTres();  
    return 0;  
}
```

Pilha de Chamadas
(Call Stack)



Definindo protótipos de funções

- A função deve estar declarada antes (acima) de ser utilizada.
 - É possível definir o protótipo das funções antes de implementá-las.

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

```
// protótipos
```

```
int max(int a, int b);
```

```
void ex02_maiorEntreTres();
```

```
int main() {  
    ex02_maiorDeTres();  
    return 0;  
}
```

```
void ex02_maiorDeTres() {  
    int a, b, c;  
    printf("Informe 3 valores: ");  
    scanf("%d %d %d", &a, &b, &c);  
    printf("MAX:%d\n", max(max(a,b), c));  
}  
  
int max(int a, int b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

Passagem de parâmetros para funções

Passagem de parâmetro

⇒ **Por valor**: parâmetro de entrada

- A função recebe uma cópia do valor passado aos parâmetros.

```
// Verifica se número é primo (true/false)  #include <stdio.h>
int isPrime(int a) {
    int div = a/2;
    while (div > 1) {
        if (a % div == 0) {
            return 0; // false
        }
        div--;
    }
    return 1; // true
}

int main() {
    printf("Prime: %d\n", isPrime(561));
    return 0;
}
```

Passagem de parâmetro

⇔ **Por endereço**: parâmetro de entrada e saída

```
// Recebe três números e retorna o maior e o menor
void minMax(int a, int b, int c,
            int* min, int* max) {
    int menor = a;
    if (b < menor) menor = b;
    if (c < menor) menor = c;

    int maior = a;
    if (b > maior) maior = b;
    if (c > maior) maior = c;

    *min = menor;
    *max = maior;
}

#include <stdio.h>

int main() {
    int x,y,z;
    printf("Informe 3 numeros: ");
    scanf("%d %d %d", &x, &y, &z);

    int m, n;
    minMax(x, y, z, &n, &m);
    printf("Menor: %d\n", n);
    printf("Maior: %d\n", m);
    return 0;
}
```

Passagem de parâmetro

⇔ **Por endereço**: parâmetro de entrada e saída

```
// Calcula as raízes do polinômio 2º grau
// Forma:  $ax^2 + bx + c = 0$ 
int polyRoots2(float a, float b, float c,
               float *root1, float *root2) {
    // 1º grau:  $bx + c = 0$ 
    if (a == 0)
        *root1 = *root2 = -c / b;
    float delta = b * b - 4 * a * c;
    if (delta < 0)
        return 0; // não possui raízes reais
    *root1 = (-b - sqrt(delta)) / (2 * a);
    *root2 = (-b + sqrt(delta)) / (2 * a);
    return 1;
}

#include <stdio.h>
#include <math.h>

int main() {
    float r1 = 0;
    float r2 = 0;

    polyRoots2(4, -10, 4, &r1, &r2);

    printf("Roots: %.2f, %.2f\n", r1, r2);
    return 0;
}
```

Adendo: problema de leitura de caracteres

- **scanf("%c")** consome um caractere do buffer de entrada
 - Quando digitamos um caractere e pressionamos ENTER, este último ficará no buffer de entrada como '\n', que alimentará automaticamente o próximo scanf("%c"), impedindo a leitura de dois caracteres em sequência.
- Duas soluções
 - **SOLUÇÃO 1: scanf(" %c") ⇒ espaço + %c**
 - O espaço antes do %c indica que o separador de entrada deve ser ignorado
 - **SOLUÇÃO 2: limpar buffer de entrada após leitura**
 - Utilizar scanf("%c") e chamar uma função para consumir o restante do buffer até '\n'

```
// utilizar após scanf ou getchar
void clearBuffer() {
    while (getchar() != '\n');
}
```

Referências

- Algoritmos e Programação
 - Marcela Gonçalves dos Santos
 - Disponível pelo Moodle
- Estruturas de Dados, Waldemar Celes e José Lucas Rangel
 - PUC-RIO - Curso de Engenharia
 - Disponível pelo Moodle
- Linguagem C, Silvio do Lago Pereira
 - USP - Instituto de Matemática e Estatística
 - Disponível pelo Moodle
- Curso Interativo da Linguagem C
 - <https://www.tutorialspoint.com/cprogramming>