



Função

Prof. Lilian Berton

São José dos Campos, 2019

Baseado no material de Ronaldo F. Hashimoto e Carlos H. Morimoto – IME/USP

Função

- Uma função é um subprograma (também chamado de sub-rotina). É um **nome dado a um trecho do algoritmo e que, em geral, encerra em si próprio um pedaço da solução de um problema maior** – o algoritmo a que ele está subordinado.
- Após sua execução, o programa volta ao ponto situado imediatamente após a chamada da função.

Exemplo lúdico 1

- Precisamos desenvolver um programa para um robô desenhar 5 retângulos, diminuindo as medidas a cada retângulo.



- ***Repita 2 vezes;***
- ***Faça Desenhe linha com a medida LadoA;***
- ***Vire à direita 90 graus;***
- ***Desenhe uma linha com a medida LadoB;***
- ***Vire à direita 90 graus.***
- ***Fim Repita***

Exemplo

- Se quiséssemos desenhar um retângulo com medidas 10cm x 5cm,
 - LadoA e LadoB seriam 10 e 5.
- A mesma sequência de comandos serviria para desenharmos um retângulo com medidas 12cm x 6cm
 - LadoA e LadoB seriam 12 e 6.
- A essas variáveis LadoA e LadoB chamamos de **parâmetros da função**. São **variáveis** que armazenam valores transferidos para a função no momento da invocação da mesma.

Exemplo

LadoA = 14

LadoB = 10

Repita 5 vezes:

Faça Função Desenhe um retângulo: LadoA, LadoB

LadoA = LadoA - 2

LadoB = LadoB - 2

Fim Repita

Função Desenhe um retângulo: LadoA, LadoB



Parâmetros

Repita 2 vezes:

Faça Desenhe linha com a medida LadoA;

Vire à direita 90 graus;

Desenhe uma linha com a medida LadoB;

Vire à direita 90 graus.

Fim Repita

Exemplo

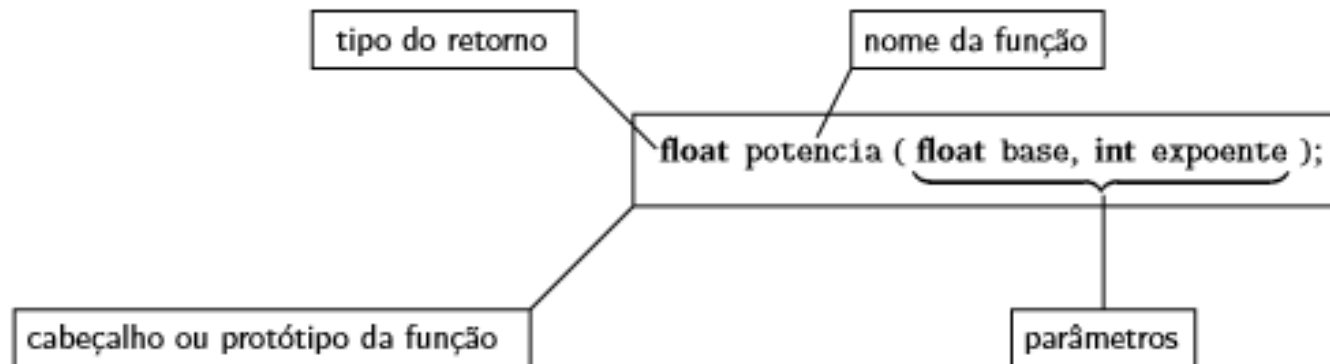


Fazendo a depuração (análise) deste pseudocódigo, teremos o seguinte resultado:

- Primeiro retângulo: Largura 14 e altura 10
- Segundo retângulo: Largura 12 e altura 8
- Terceiro retângulo: Largura 10 e altura 6
- Quarto retângulo: Largura 8 e altura 4
- Quinto retângulo: Largura 6 e altura 2

Funções em C

- Um programa na linguagem C pode ser organizado na forma de **funções**, onde cada função é responsável pelo cálculo / processamento de uma parte do programa.
- Em particular, **o main é uma função!** Todo programa em C precisa de uma função chamada main (função principal), que, dentre todas as funções que estão em seu programa, **ela é a primeira a ser executada**.
- O cabeçalho ou protótipo de uma função contém **o tipo** do valor devolvido (int, char ou float), o **nome** da função e **uma lista de parâmetros**, entre parênteses, seguido de um ponto e vírgula dispostos da seguinte forma:



Retorno das funções

- Uma função sempre deve ter um retorno de acordo com o tipo especificado no cabeçalho:

```
float soma (float a, float b) {  
...  
return x; //x deve ser float  
}
```

```
int calculo () {  
...  
return y; //y deve ser int  
}
```

- Para uma função não retornar nada use o tipo void:

```
void leitura (char a) {  
....  
// não precisa ter retorno  
}
```


Retorno das funções

- **O valor da variável retornado pela função deve ser recebido de volta por outra variável que chamou a função!**

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

```
#include<math.h>
```

```
int potencia();
```

```
int main() {
```

```
    int pot = potencia();           // pot recebe o resultado da função e deve ser int
```

```
    printf("O resultado eh %d",pot);
```

```
}
```

```
int potencia() {
```

```
    int x,p,;
```

```
    scanf("%d %d",&x,&p);
```

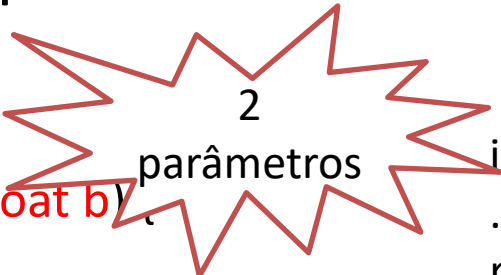
```
    return pow(x,p);               //o retorno deve ser um inteiro
```

```
}
```

Parâmetro das funções

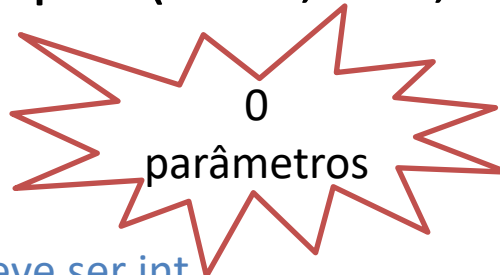
- Uma função pode receber parâmetros ou não, os parâmetros podem ser de qualquer tipo (float, int, char, etc):

float soma (float a, float b) {
...
return x; //x deve ser float
}



A red starburst shape with the number '2' inside, and the word 'parâmetros' below it, pointing to the two parameters 'float a' and 'float b' in the function signature.

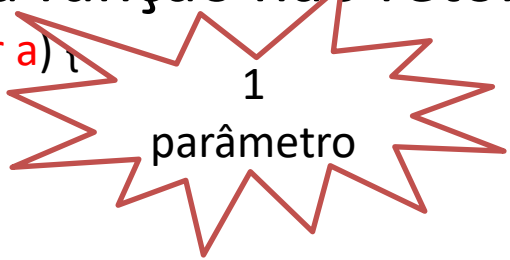
int calculo () {
...
return y; //y deve ser int
}



A red starburst shape with the number '0' inside, and the word 'parâmetros' below it, pointing to the empty parameter list in the function signature.

- Para uma função não retornar nada use o tipo void:

void leitura (char a) {
....
}



A red starburst shape with the number '1' inside, and the word 'parâmetro' below it, pointing to the parameter 'char a' in the function signature.

Parâmetro de funções

- Posso alterar o nome dos parâmetros na função:

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

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

```
int main() {  
    int x, y, s;  
    scanf("%d %d",&x,&y);  
    s = soma(x,y);  
    printf("A soma dos valores eh %d",s);  
}
```

// s recebe o resultado da função

```
int soma(int a, int b) {  
    return (a+b);  
}
```



Uma cópia dos valores de x,y é feito em a,b

// o retorno deve ser inteiro

Parâmetro de funções

- **As variáveis declaradas em cada função são independentes umas da outra e são desalocadas da memória após o término da função. Assim, posso ter variáveis com o mesmo nome em funções diferentes.**

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

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

```
int main() {  
    int x, y, s;  
    scanf("%d %d",&x,&y);  
    s = soma(x,y);    // s recebe o resultado da função  
    printf("A soma dos valores eh %d",s);  
}
```

```
int soma(int a, int b) {  
    int s;  
    s = (a+b);  
    return s;    //s deve ser int  
}
```

Exemplo 1

- Você pode deixar o código das funções antes da main.
- Mas o mais indicado é inserir as funções depois da main. Nesse caso será necessário incluir o protótipo delas embaixo das bibliotecas.

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

```
float soma (float a, float b);
```

➡ Protótipo da função

```
int main() {
```

```
    float a, b, s;
```

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

```
    s = soma(a,b);
```

```
    printf("O resultado da soma eh: %f",s);
```

```
    return 0;
```

```
}
```

➡ A variável s recebe o valor retornado pela função soma

```
float soma (float a, float b) {
```

```
    float s;
```

```
    s = a+b;
```

```
    return x; //x deve ser float
```

```
}
```

➡ A função recebe dois parâmetros , a e b

➡ A função retorna um valor para o local que a chamou

Exemplo 2

- Faça duas funções que retorne o menor e o maior valor entre dois números lidos;

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

```
float menor2(float num1, float num2);
```

```
float maior2(float num1, float num2);
```



Protótipo da função

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

```
float n1,n2, menor, maior;
```

```
scanf("%f %f", &n1,&n2);
```

```
menor = menor2(n1,n2);
```

```
maior = maior2(n1,n2);
```

```
}
```



As variáveis menor e maior recebem o valor retornado pelas funções menor2 e maior2

```
float menor2(float num1, float num2) {
```

```
if(num1 <= num2)
```

```
    return num1;
```

```
else return num2;
```

```
}
```



A função retorna o menor valor para o local que a chamou

```
float maior2(float num1, float num2) {
```

```
if(num1 <= num2)
```

```
    return num2;
```

```
else return num1;
```

```
}
```



A função retorna o maior valor para o local que a chamou

Exemplo 2

- Faça duas funções que retorne o menor e o maior valor entre dois números lidos;

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

```
float menor2(float num1, float num2);
```

```
float maior2(float num1, float num2);
```

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

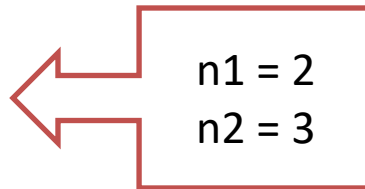
```
float n1,n2, menor, maior;
```

```
scanf("%f %f", &n1,&n2);
```

```
menor = menor2(n1,n2);
```

```
maior = maior2(n1,n2);
```

```
}
```



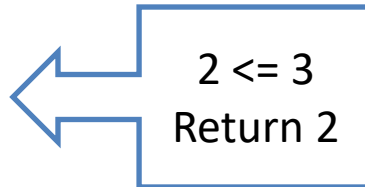
```
float menor2(float num1, float num2) {
```

```
if(num1 <= num2)
```

```
    return num1;
```

```
else return num2;
```

```
}
```



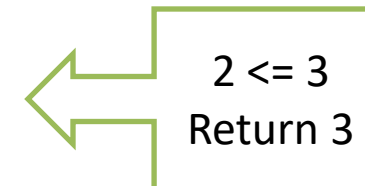
```
float maior2(float num1, float num2) {
```

```
if(num1 <= num2)
```

```
    return num2;
```

```
else return num1;
```

```
}
```



Passando vetor como parâmetro em função

```
#include <stdio.h>
float soma (float V[ ]);

int main() {
    float V[10], s;
    for(int i = 0; i < 10; i++) {
        scanf("%f ", &V[i]);
    }
    s = soma(V);
    printf("O resultado da soma eh: %f",s);
return 0;
}

float soma (float V[ ]) {
    float x;
    for(int i = 0; i < 10; i++) {
        x = x + V[i];
    }
    return x; //x deve ser float
}
```


Funções

- **Mecanismo do processo de chamada de função**
 1. Valor dos parâmetros/argumentos é calculado pelo programa que está chamando a função.
 2. Sistema cria novo espaço para todas as variáveis locais da função (estrutura de pilha).
 3. Valor de cada parâmetro/argumento é copiado na variável parâmetro correspondente na ordem em que aparecem.
 4. Comandos do corpo da função são executados até:
 - 4.1 Encontrar comando return.
 - 4.2 Não existirem mais comandos para serem executados.
 5. O valor da expressão return, se ele existe, é avaliado e retornado como valor da função.
 6. Pilha de variáveis criada é liberada.
 7. Programa que chamou continua sua execução.

Vantagens de funções

- **Projeto top-down**
- Funções permitem dividir um programa em pedaços menores.
- Facilita sua leitura.
- Melhor estratégia para escrever programas é começar com o programa principal (main).
- Pensar no programa como um todo e depois identificar as principais partes da tarefa completa:
 - Maiores pedaços são candidatos a funções;
 - Mesmo essas funções podem ser decompostas em funções menores;
 - Continuar até cada pedaço ser simples o suficiente para ser resolvido por si só;

Exercícios

1. Elabore uma função que receba um vetor contendo N valores e um valor X escolhidos pelo usuário, retorne para a main o número de vezes que esse elemento ocorreu no vetor.

2. Faça um programa que leia o total gasto pelo cliente de uma loja. Mostre as seguintes opções de pagamento, solicite a opção desejada e imprime o valor total do pagamento.
 - 1) Opção: a vista com 10% de desconto
 - 2) Opção: em duas vezes (preço da etiqueta)
 - 3) Opção: de 3 até 10 vezes com 3% de juros ao mês (somente para compras acima de R\$ 100,00).

OBS: fazer uma função para cada opção. E imprimir na main o valor final da compra.



FRONT END



BACK END



WEEK END