Fundamentos de Programação

Funções

Prof. Luiz Fernando Carvalho

luizfcarvalho@utfpr.edu.br

Funções

- Em C, uma função é um pedaço de código, dentro de um programa maior, que realiza uma certa tarefa com uma certa independência do resto do programa

Todo programa possui pelo menos uma função: main

Funções – Porque utilizar

- Clareza do código
  - Separando pedaços de código da função `main()`, podemos entender mais facilmente o que cada parte do código faz;
- Reutilização
  - Quando se precisa executar certa tarefa várias vezes ao longo do programa. Utilizando funções não é necessário repetir todo o código várias vezes, além de facilitar a manutenção do código, pois a correção de erros será somente em um lugar;
- Independência
  - Uma função é relativamente independente do código que a chamou. Uma função pode modificar variáveis globais, mas limitando-se aos dados fornecidos pela chamada de função.

Função

- Em C, uma função deve ter as seguintes características:
  - Um nome pela qual ela possa ser chamada
    - Os nomes possíveis seguem as mesmas restrições que os nomes de variáveis;
- Valores de entrada ou parâmetros
  - São os valores sobre os quais a função pode operar;
  - Os parâmetros das funções atuam de maneira análoga às variáveis das funções matemáticas;
  - Também é possível criar funções sem parâmetros;
- Um tipo de retorno
  - Corresponde ao “resultado” da função;
  - Também é possível criar funções que não devolvem nenhum valor de retorno

Função

- Para definir uma função, usamos a seguinte estrutura:

```
tipo_retorno nome_da_função(tipo parâmetro_1, …, tipo parametro_n){
    //conteúdo da função;
}
```

- O `tipo_retorno` pode ser qualquer um dos tipos usados para variáveis;
- No caso em que não existe valor de retorno, é possível usar no lugar do `tipo_retorno` a palavra **`void`** (vazio, em inglês)
- Ela apenas indica a ausência de um valor;
- O conjunto dessas três definições (tipo, nome, parâmetros) é chamado de cabeçalho/assinatura da função.

Exemplo

- Uma função que calcula a soma dos divisores de um número inteiro n:

```
int soma_divisores(int n)
```

- Como entrada, tem-se o número *n,* que será uma variável do tipo **`int`**;

- Como saída, teremos outro valor do tipo **`int`**, que corresponderá a soma dos divisores de n

Exemplo

- Uma função que recebe os valores correspondentes de um mês e um ano e imprime o calendário desse mês

```
void imprime_calendario(int mes, int ano)
```

- Nessa caso, não há nenhum valor de saída (os dados são enviados diretamente para a tela, com a função **`printf`**)

- Isso indica que a palavra **`void`** será usada no lugar do tipo do retorno

Exemplo

- Uma função que recebe dois inteiros, a e b, devolve o valor da potência

```
int potencia(int a, int b)
```

- Novamente, todos os valores envolvidos são do tipo **`int`**

```
int potencia(int a, b) --------►
```

Errado: O tipo de cada parâmetro deve ser especificado

Usando Funções

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

void mensagem(){
    printf("Exemplo de uma funcao simples!\n");
}

int main(){
    printf("O exemplo de chamada de funcao!\n");
    mensagem(); //chamada da função

    return 0;
}
```

Quando ocorre a chamada da função, a execução da função `main` é suspensa até a execução completa da função mensagem. Após isso, a `main` continua de onde parou.

Parâmetros da Função

- Uma vez definidos os parâmetros no cabeçalho da função, pode-se acessá-los como se fossem variáveis normais
- Por exemplo, uma função que recebe dois inteiros e imprime sua soma na tela:

```
void imprime_soma(int a, int b){
    int soma;
    soma = a + b;
    printf("%d\n", soma);
}
```

```
void imprime_soma(int a, int b){
    printf("%d\n", a + b);
}
```

- A função não tem nenhum resultado a devolver para o programa
  - Portanto, usa-se a palavra `void` para o "tipo" de saída

O conteúdo de uma função deve ser construído como de costume dentro da função `main`

Usando Funções

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

void imprime_soma(int a, int b){
    int soma;
    soma = a + b;
    printf("%d\n", soma);
}

int main(){

    imprime_soma(3, 5); //chamada da função

    return 0;
}
```

Retorno da Função

- Para devolver o valor de saída, usa-se o comando return seguido do valor de saída;
- O valor pode ser qualquer expressão que seja legítima
  - de se colocar no lado direito de uma atribuição;
  - O valor de uma variável;
  - Uma constante numérica ou caractere;
  - Uma expressão aritmética;
  - Um vetor ou matriz;
  - Etc.

```
return 0;
return x;
return x * x;
return y + 1;
```

Retorna o resultado do cálculo

Na linguagem C não é possível retornar mais de um valor a partir de uma função usando o comando `return`

Retorno da Função

- A função a seguir devolve para o programa a soma dos dois números recebidos como parâmetros e retorna o resultado:

```
 1  float soma(float a, float b){
 2     return a + b;
 3  }
 4
 5  int main(){
 6     float resultado;
 7
 8     resultado = soma(3.5, 8.0);
 9     printf("A soma e' %f", resultado);
10
11     printf("A soma e' %f", soma(5.0, 2.5));
12
13     return 0;
14  }
```

Retorno da Função

- É importante ressaltar que a instrução `return` também encerra a execução da função
  - O programador deve usar esse comando somente quando não houver mais nada a fazer dentro da função;
- Se o comando `return` for colocado no meio da função, ela devolverá o valor indicado e ignorará todo o resto da função

```
float divisao(float a, int b){
//calcula a divisão a/b
     if(b == 0)
        return 0;

     return a/b;
}
```

Se o valor de b for zero, a função é encerrada neste ponto

Se o **return** anterior não for executado, este encerrará a execução da função

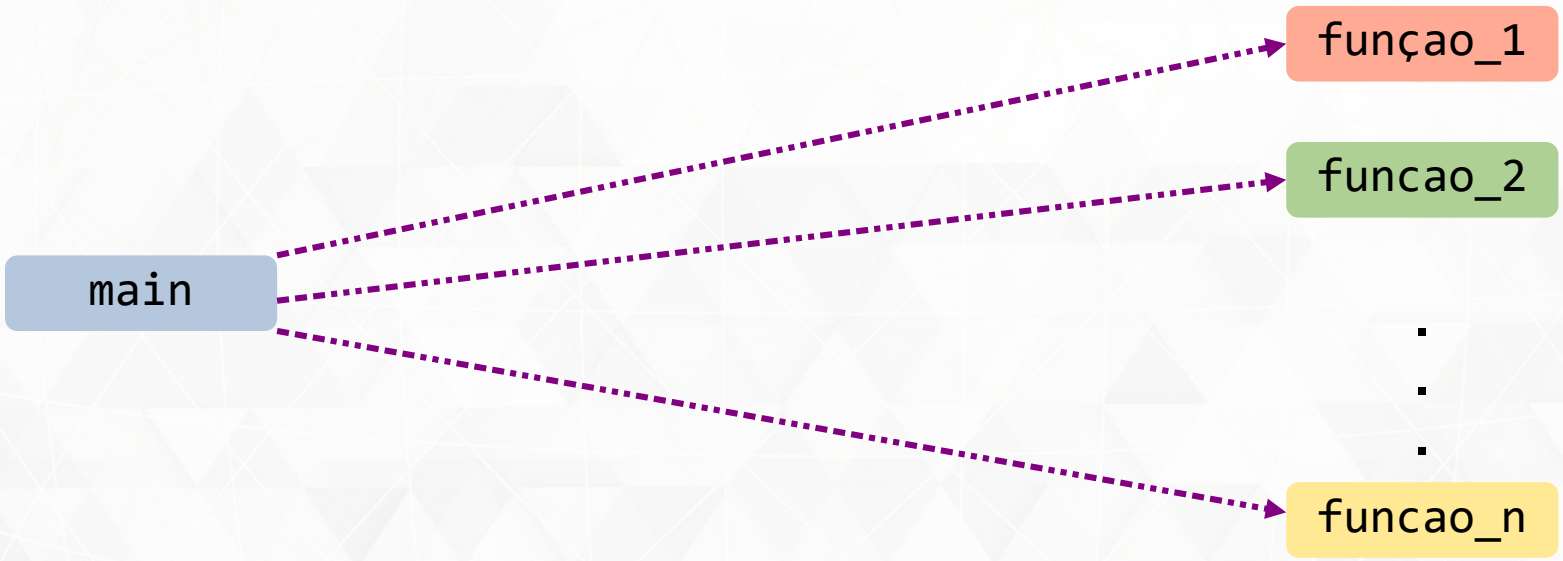
Retorno da Função

- Vale também salientar que uma função `void` **NÃO PODE** devolver nenhum valor no `return`
- **ATENÇÃO:** O comando `return` **NÃO É NECESSÁRIO** em funções com tipo de retorno `void`!
- No entanto, **PODE-SE** usar a instrução `return` (sem nenhum valor) para terminar uma função `void`

```
void imprime_numero(int n)
{
      if(n < 0) {
         printf("Não quero imprimir números negativos!\n");
         return;
      }
      printf("%d\n", n);
}
```

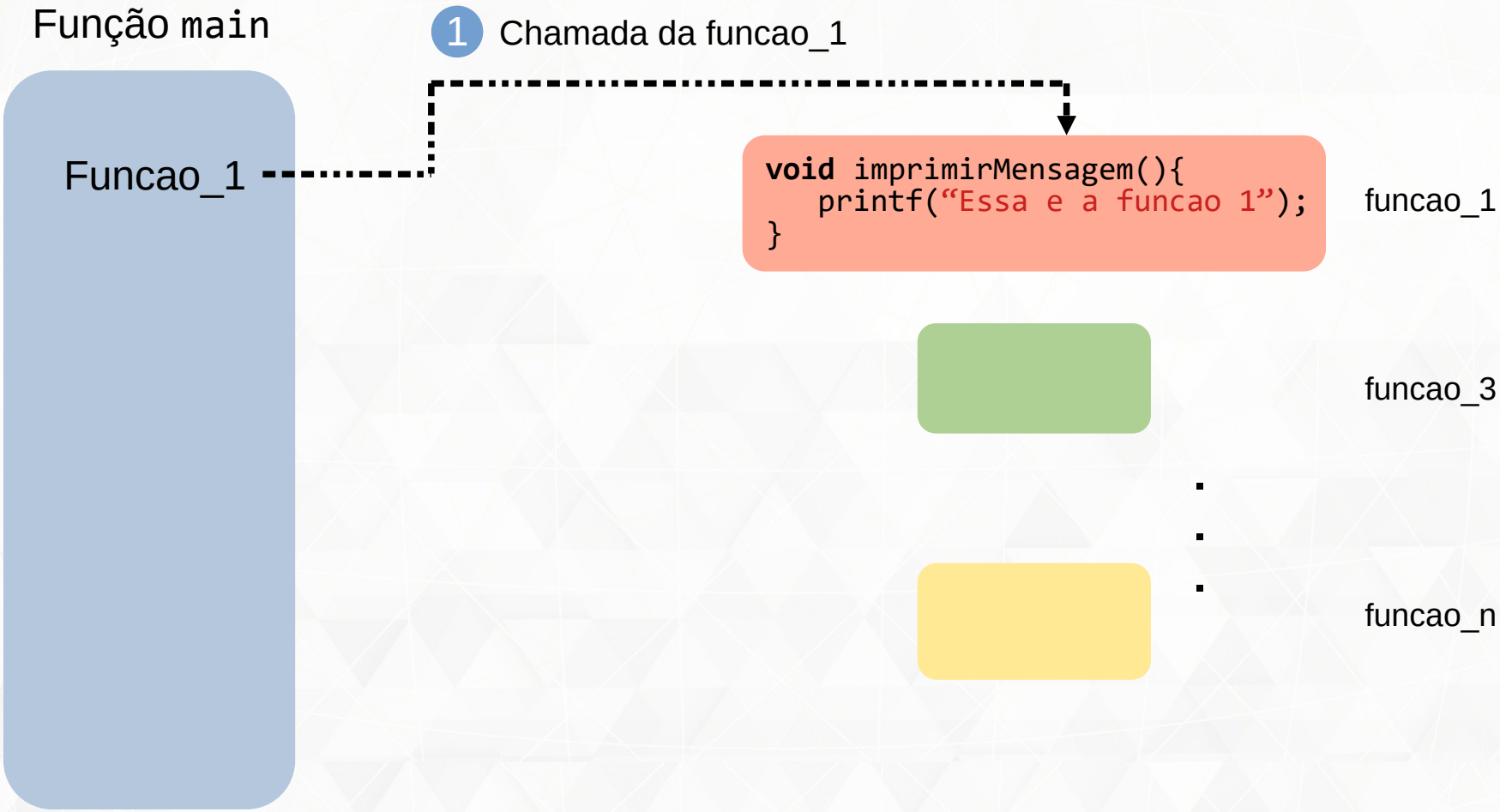
Usando Funções

- **Uma boa prática:** função main somente organiza as **chamadas** das demais funções



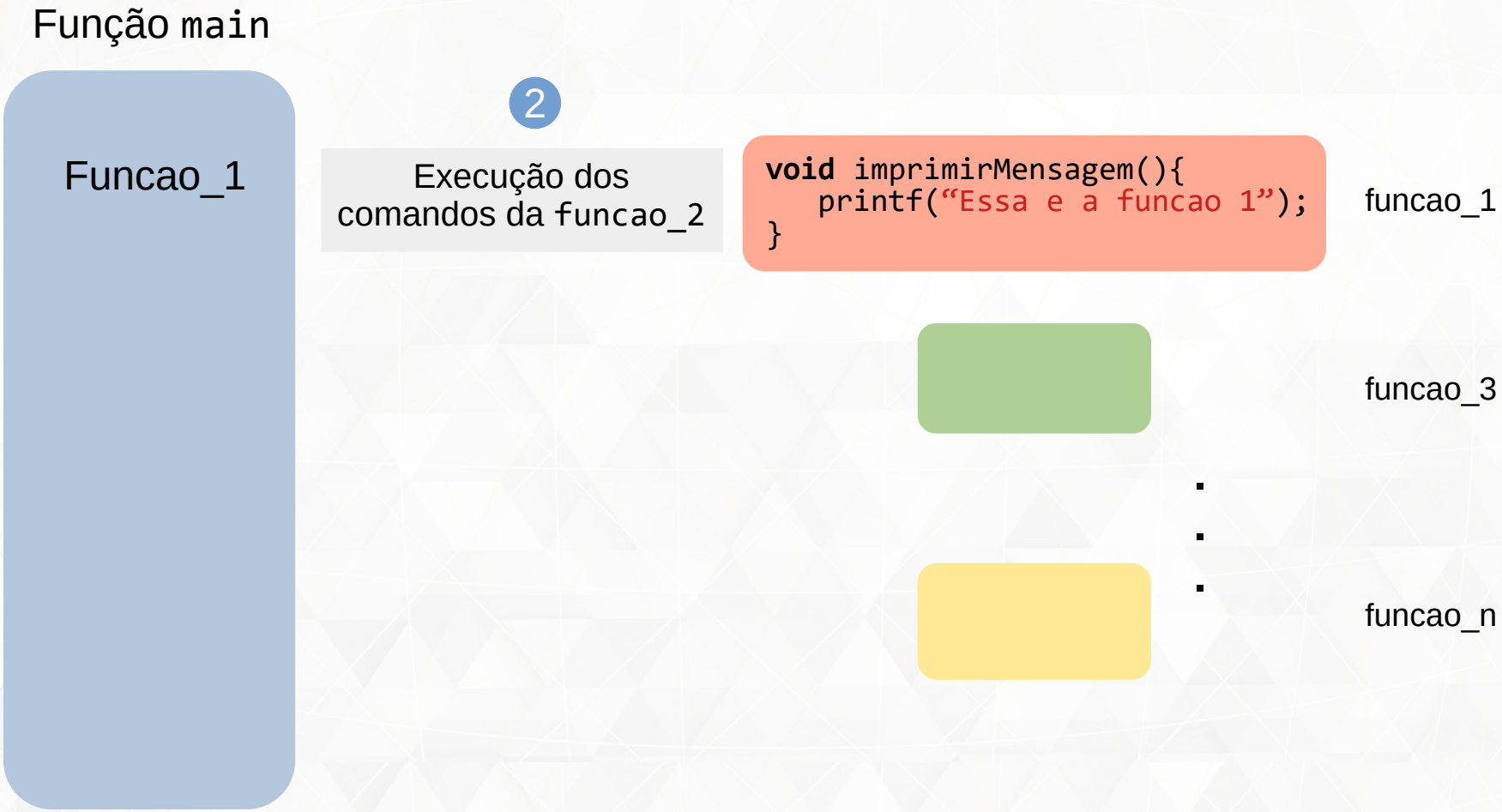
Porque essa é uma boa prática?

Usando Funções



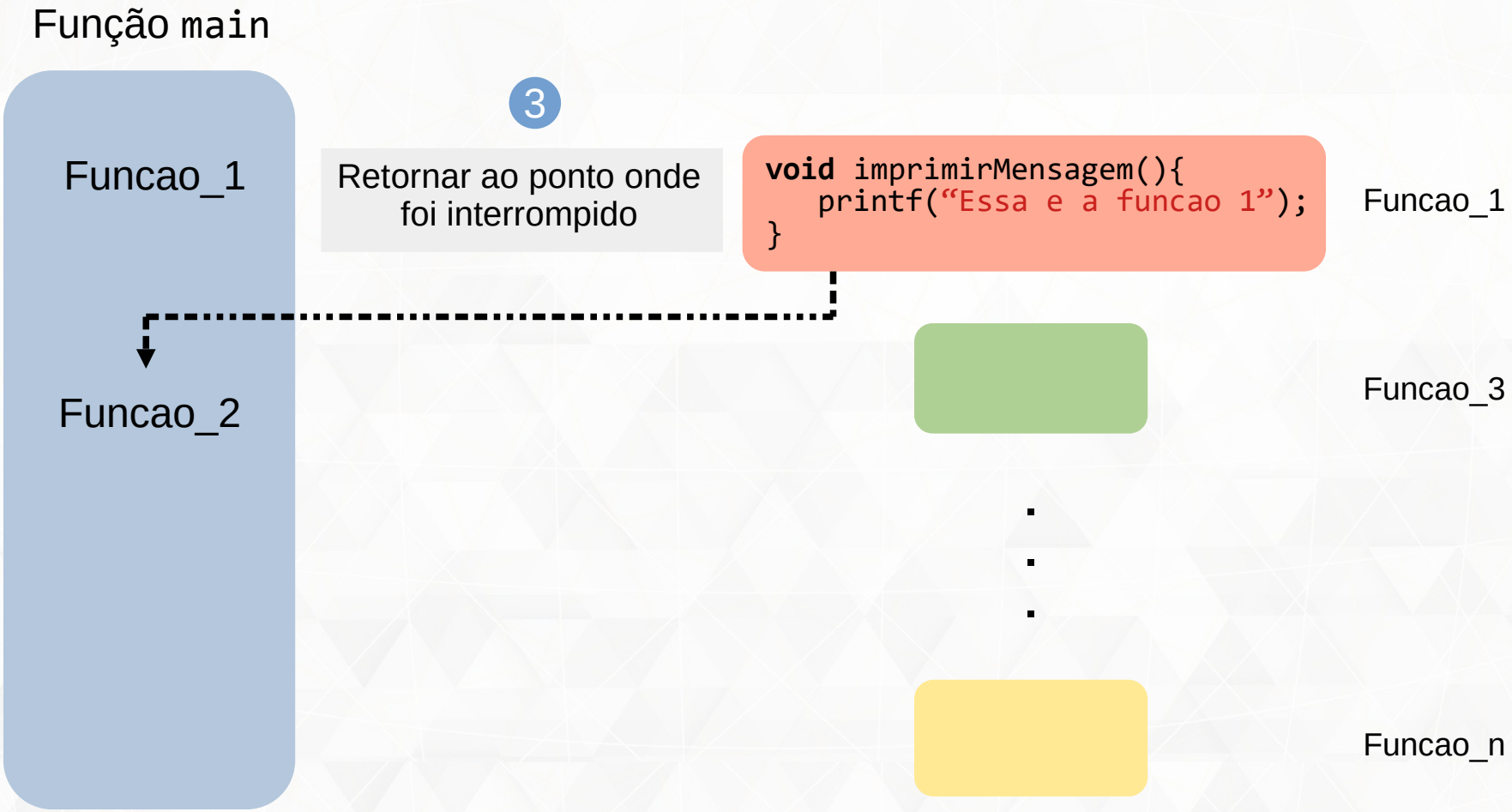
Porque essa é uma boa prática?

Usando Funções



Porque essa é uma boa prática?

Usando Funções



Porque essa é uma boa prática?

Definição de Funções

```
float divide(float dividendo, float divisor){
    return dividendo/dividor;
}
```

Definição da função

```
int main(){
    float dividendo = 10;
    float divisor = 3;
    float resultado;

    resultado = divide(dividendo, divisor);
    printf("%f", resultado);

    return 0;
}
```

Função main

Escopo de Variáveis

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

//declaração de variáveis globais

void funcao(parâmetros){ //parâmetros têm escopo local
   // declaração das variáveis locais da função
}


int main(){
  //declaração das variáveis locais da main()

   return 0;
}
```

Escopo de Variáveis

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

void funcao1(int a, int b){
    int soma;
    soma = a + b;
}

int main(){
   int numero1, numero2;
   numero1 = 2;
   numero2 = 8;

   funcao1(numero1, numero2);
   printf("A soma e': %d", soma);

   return 0;
}
```

A variável **soma** só existe dentro da **funcao1**

Qual o valor será impresso?

Nenhum!!!

Esse comando resultará em um erro

Exercícios

1) Escreva uma função que recebe por parâmetro as 3 notas de um aluno e uma letra. Se a letra for A, o procedimento calcula a média aritmética das notas do aluno, se for P, a sua média ponderada (pesos: 5, 3 e 2) e se for S, a soma das notas. O valor calculado também deve ser mostrado dentro da função;

2) Faça uma função que recebe a média final de um aluno por parâmetro e retorna o seu conceito, conforme a tabela abaixo:

Nota	Conceito
[0, 5[	D
[5, 7[	C
[7, 9[	B
[9, 10]	A

Exercícios

3) Crie uma função que que receba o valor de um inteiro positivo N, calcule e retorne o fatorial desse número.

4) Escreva uma função que recebe por parâmetro um valor inteiro e positivo N e retorna o valor de S.

$$S = \frac{2}{4} + \frac{5}{5} + \frac{10}{6} + \ldots + \frac{N^2 + 1}{N + 3}$$

Exercícios

5) Você possui um pedaço retangular de papelão como mostra a Figura 1(a) e deve fazer uma caixa com ele. Para isso, vai cortar quadrados nos cantos (Figura 1(b)) e dobrar as laterais (Figura 1(c)), formando uma caixa (aberta em cima)

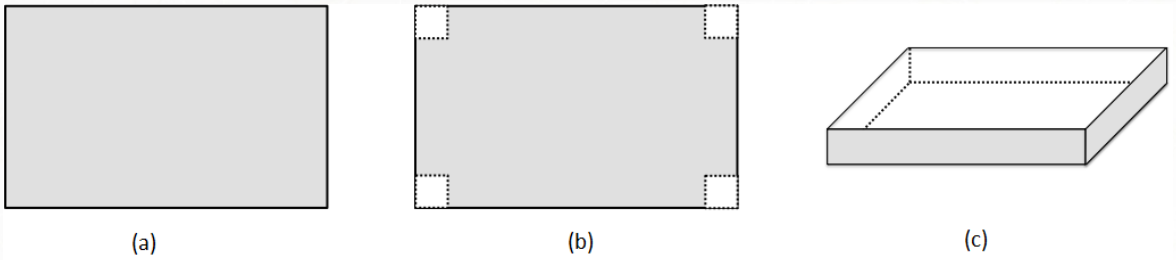


Figura 1

- Os cantos cortados são quadrados do mesmo tamanho. A dúvida é: qual o tamanho dos lados desses cortes quadrados para que a caixa tenha o maior volume possível?

- Por exemplo, com um pedaço de papelão de 25 x 40 cm (Figura 2(a)), o ideal é cortar quadrados de 5 cm (Figura 2-(b)), e assim obter uma caixa de volume 2250 cm3 (Figura 2-(c)).

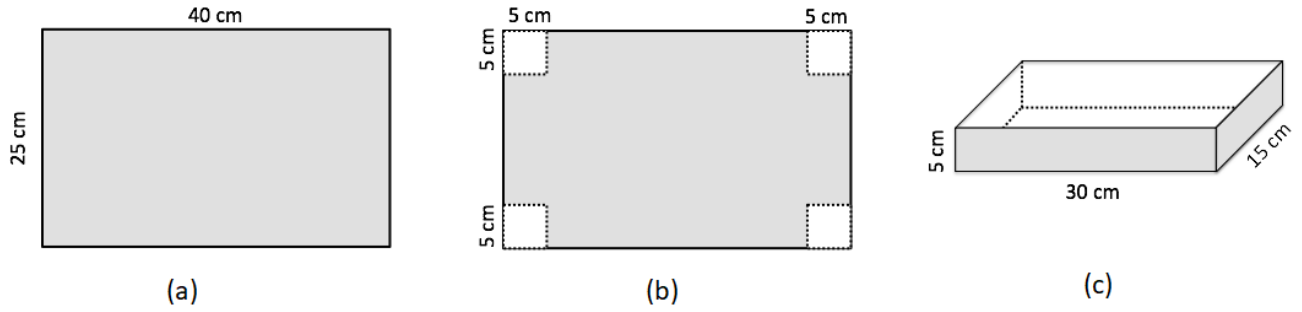


Figura 2

Exercícios

6) Faça uma função que calcule e retorne o número neperiano `e`, `e=2,71828183`, usando a série a seguir:

$$e = \sum_{n=0}^{N} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} \cdots$$

A função deve ter como parâmetro o número de termos que serão somados, `N`. Note que quanto maior esse número, mais próxima do valor `e` está a resposta.