funções

funções

tópicos

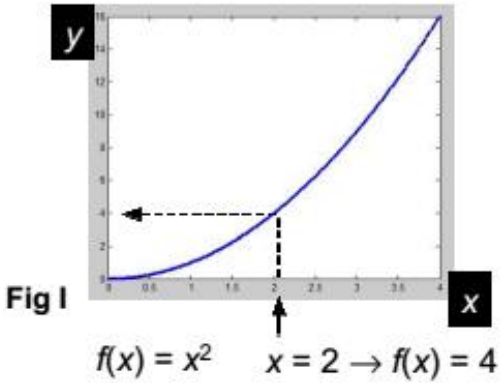
- o que é uma função
- como definir uma função
- assinatura (protótipo) de uma função
- pilha de execução
- escopo de variáveis

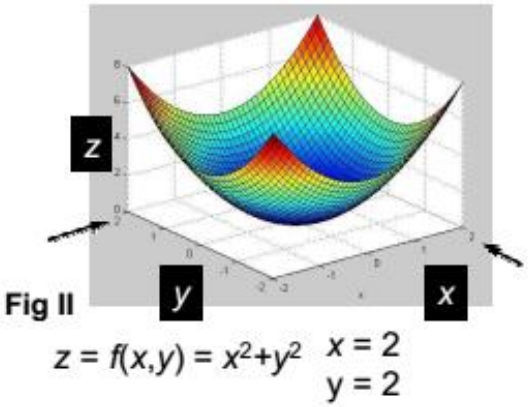
referência

- Capítulo 3 da apostila

o que são funções?

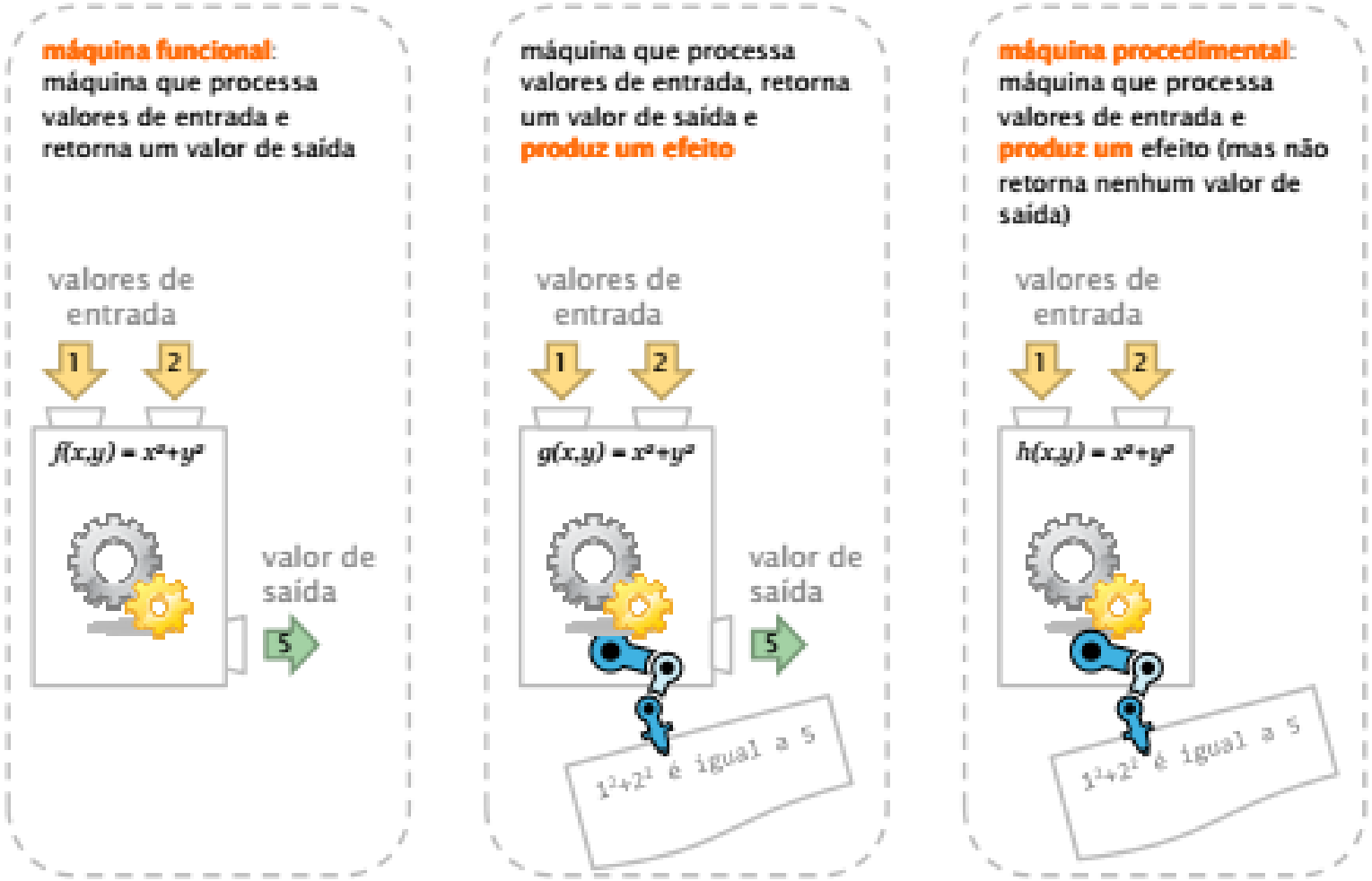
funções matemáticas





- **Função** é uma relação de um ou vários valores de argumentos de entrada e um **ÚNICO** resultado de saída
- O conjunto de todos os valores de entrada de uma função é chamado de **domínio** da função.
- O conjunto de todos os resultados é chamado de **imagem** da função.
- Normalmente, a imagem é um subconjunto de um conjunto de resultados possíveis, chamado de **contradomínio**.

funções e procedimentos



Em C, funções e procedimentos são chamados simplesmente de **funções**.

funções e procedimentos

- Em C, um procedimento (retornando ou não valores) continua sendo chamado de função (*function*).
- “Funções” que não retornam valor não podem ser usadas em expressões ou como argumento de outras funções.
- Em C, o comando **return** define o que a função retorna.

funções

- Uma **função** é um pedaço de código que você escreve **uma vez**, mas pode **usar em várias partes** do seu código.
- Um **parâmetro** é uma variável com valor já atribuído quando a função começa. O usuário da função **deve prover** os valores dos **parâmetros** quando ele chama a função.

por que usar funções?

- Funções nos permitem desenvolver programas como **uma sequência de subpassos** (Cada sub-passo pode ser sua uma função)
  - Quando um problema parece ser muito difícil, dividí-lo em suproblemas pode ajudar muito na solução.
- Funções nos permitem **reusar** código em vez de reescrevê-los.
- Funções nos permitem **manter o espaço de nomes de variáveis limpo** (variáveis locais apenas "vivem" enquanto a função "vive").
  - Uma função 1 pode usar uma variável cahamada i, e uma função 2 também pode usar uma variável chamada i; não ocorrerá nenhuma confusão. Cada variável i só existe quando o computador está executando a dada função.
- Funções nos permitem **testar pequenas partes** dos nossos programa de forma isolada do resto.

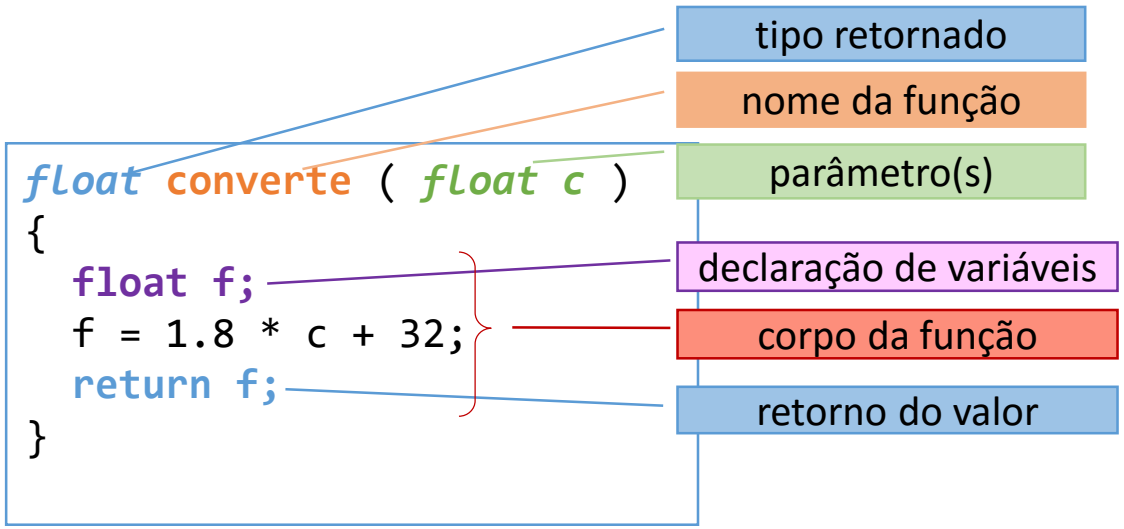
usando (“chamando”) funções

```
#include <stdio.h>
int main (void)
{
  float nota1, nota2, nota3;
  float media;

  nota1 = le_nota();
  nota2 = le_nota();
  nota3 = le_nota();

  media = calc_media (nota1, nota2, nota3);

  imprime_media(media);

  return 0;
}
```

As funções le_nota, calc_media e imprime_media precisam ser definidas pelo programador.

como definir uma função

```
tipo_retornado nome_da_função ( lista de parâmetros ... )
{
    corpo_da_função
}
```



Sintaxe
(forma da função)

semântica
(funcionalidade ou comportamento; o que a função faz).

funções “equivalentes” para o cálculo da média

```c
float calc_media ( float a, float b, float c )
{
  float soma, media;
  soma = a + b + c;
  media = soma / 3.0;
  return media;
}
```

```c
float calc_media ( float a, float b, float c )
{
  float soma;
  soma = a + b + c;
  return soma/3.0;
}
```

```c
float calc_media ( float a, float b, float c )
{
  return (a + b + c) / 3.0;
}
```

chamada da função

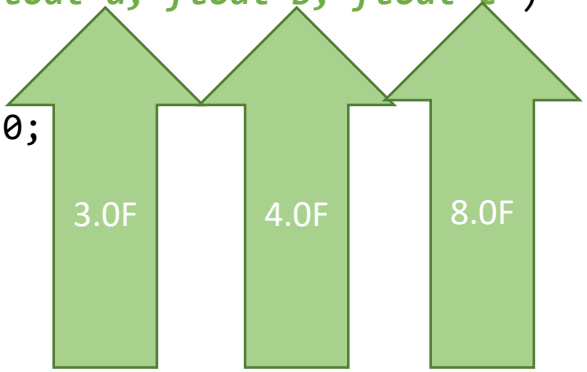
```c
int main (void) {
  float media = calc_media (3.0F, 4.0F, 8.0F);
  printf (“A media e: \n”, media)
  return 0;
}
```

printf também é uma função

o que acontece quando uma função é chamada?

1. Ao chamar a função **calc_media**, os valores passados como parâmetros (3.0F, 3.0F e 8.0F) são **copiados** para os **parâmetros da função**, na respectiva ordem.

```
float calc_media (   float a, float b, float c )
{
  float res;
  res = (a + b + c)/3.0;
  return res;
}

int main(void) {
  float media;
  media = calc_media (   3.0F,      4.0F,    8.0F);
  printf ("A media e: %.2f.\n", media);
  return 0;
}
```



2. A função **calc_media** executa: `res = (a + b + c)/3.0;`

3. A função calc_media retorna o valor na variável **res** (do tipo **float**) para a função que a chamou:

`return res;` → `return 5.0F;`

4. Na função main, o valor retornado é atribuído à variável media:

`media = calc_media (3.0F, 4.0F, 8.0F);` → `media = 5.0F;`

5. O restante do programa é executado, imprimindo a saída.

usando funções em expressões

- É possível chamar funções em expressões, desde que o tipo de retorno seja **compatível:**

```
media = ( le_nota() + le_nota() + le_nota() ) / 3.0;

...

if (le_nota () > 3) {
  ...
}

...

while ( le_nota() >= 0 ) {
  ...
}

...
```

usando funções em expressões

- É possível chamar uma função e passar o resultado desde para outra função:

```
int main (void) {
  float nota1, nota2, nota3, media;
  nota1 = le_nota();
  nota2 = le_nota();
  nota3 = le_nota();
  media = calc_media (nota1 + nota2 + nota3) / 3.0;
  imprime_media (media);
  return 0;
}
```

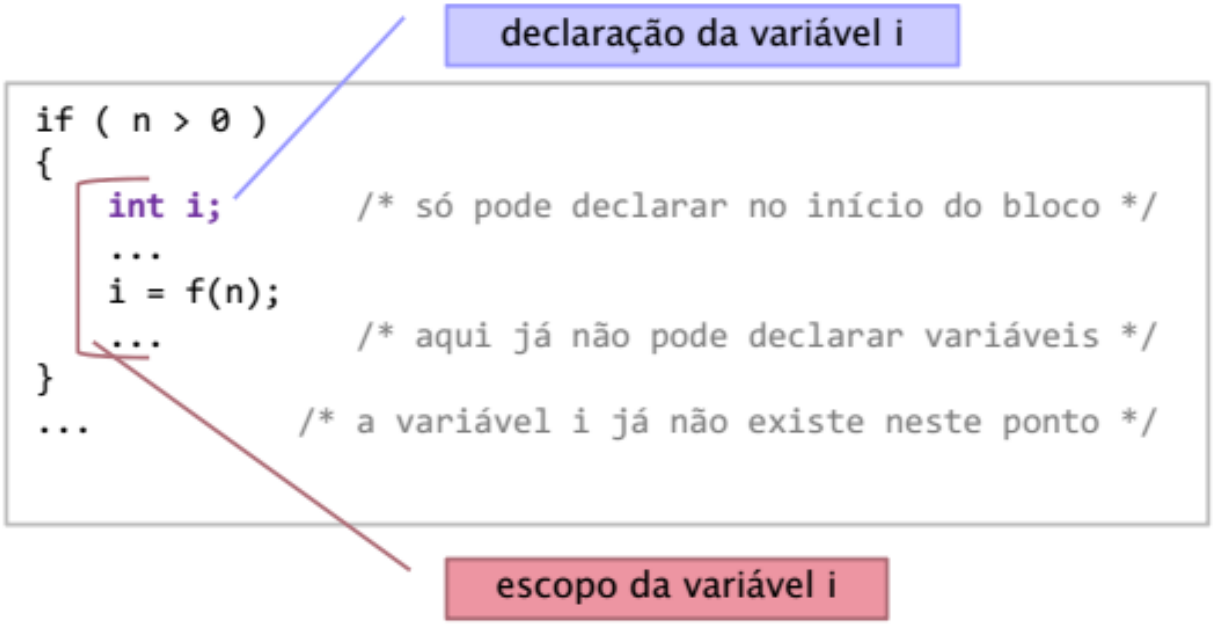
```
int main (void) {
  float media;
  media = calc_media (le_nota(), le_nota(), le_nota());
  imprime_media (media);
  return 0;
}
```

```
int main (void) {
imprime_media ( calc_media (le_nota(), le_nota(), le_nota()) );
  return 0;
}
```

escopo de uma variável

variável local

escopo é o bloco em que a variável foi declarada



tipos de variáveis quanto ao escopo

tipo	declaração	declarada onde	escopo (acessível de onde)
global	int a;	fora de qualquer função (início do programa)	todos os módulos do programa (outros módulos devem declará-la como externa); valor é sempre mantido
local (ou automática)	int a;	dentro de um bloco	somente no bloco; valor não é mantido
estática em função	static int a;	dentro de uma função	alocada em área fixa da memória; valor é mantido durante toda a execução do programa
estática no módulo	static int a;	fora de qualquer função	apenas no módulo em que é declarada
externa	extern int a;	fora de qualquer função, fazendo referência à variável global correspondente declarada em outro módulo	todos os módulos do programa em que ela esteja declarada