

**INSTITUTO FEDERAL**  
Sul-rio-grandense

Câmpus  
Charqueadas

EDUCAÇÃO  
**PÚBLICA**  
**100%**  
GRATUITA

# Funções em C

Programação Estruturada

Prof. André del Mestre

# Principais Conceitos

# Funções

## Forma Geral

- `tipo_de_retorno` - valor que a função pode retornar
  - Função sem retorno: void
  - Função com um retorno, escolha: float, char, int, ...
- `nome_da_funcao` - segue as mesmas regras de declaração de variáveis
  - Iniciar com letra, não pode usar caracteres especiais, palavras-chave do sistema, etc...
- `lista_de_parametros` - lista de valores passados para função processar
  - Pode-se definir quantos parametros quiser
- `secao de comandos` - descrição da lógica da função
  - Use operadores lógicos, aritméticos, chamadas de funções, estruturas de seleção e repetição

```
tipo_de_retorno nome_da_funcao(lista_de_parametros) {  
    //secao de comandos  
}
```

# Funções

## Lista de Parametros

- `lista_de_parametros` - Forma Geral
  - `tipo1 nome1, tipo2 nome2, ..., tipoN nomeN`
- Os nomes dos parametros seguem as mesmas regras de declaração de variaveis
  - Iniciar com letra, nao pode usar caracteres especiais, palavras-chave do sistema, etc...

```
//CERTO
float soma(float a, float b) {
    return a+b;
}

//ERRADO
float soma(float a, b) {
    return a+b;
}
```

# Funções

## Retorno de Funções

- `tipo_de_retorno` - Função pode retornar ou não um valor
  - Função sem retorno: `void`
  - Função com retorno: `char`, `float`, `int`, etc...
    - OBRIGATORIO comando `return`

```
//exemplo de retorno void!
void printSoma(float a, float b){
    printf("%i+%i=%i\n", a, b, a+b);
    //nao use return em funcao void!
}
```

```
//exemplo de retorno diferente de void!
float soma(float a, float b){
    float res = a+b;
    return res;
}
```

# Funções

## Comando return

- O valor gerado pela função é dado pelo comando **return**
  - **return** valor, variável ou expressão;
- O **return** encerra a execução de qualquer função
- Preocupe-se com a compatibilidade de tipos ao retornar funções

```
float exemplo(float a, float b){  
    float res = a+b;  
    if(a>b){  
        return res;           //variavel  
    }else if(a<b){  
        return b-a;           //expressao  
    }else{  
        return 1.5;           //valor  
    }  
    printf("Esta msg nunca sera executada");  
}
```

# Funções

## Passagem de parametro

- Saber usar a função e fundamental
  - Função soma () recebe dois parametros float e retorna um float

```
float soma(float a, float b) {  
    return a+b;  
}  
int main() {  
    float res, y=9.0;  
    res=soma(2.0, y);  
    printf("A soma eh %f\n", res);  
    return 0;  
}
```

Retornou 11

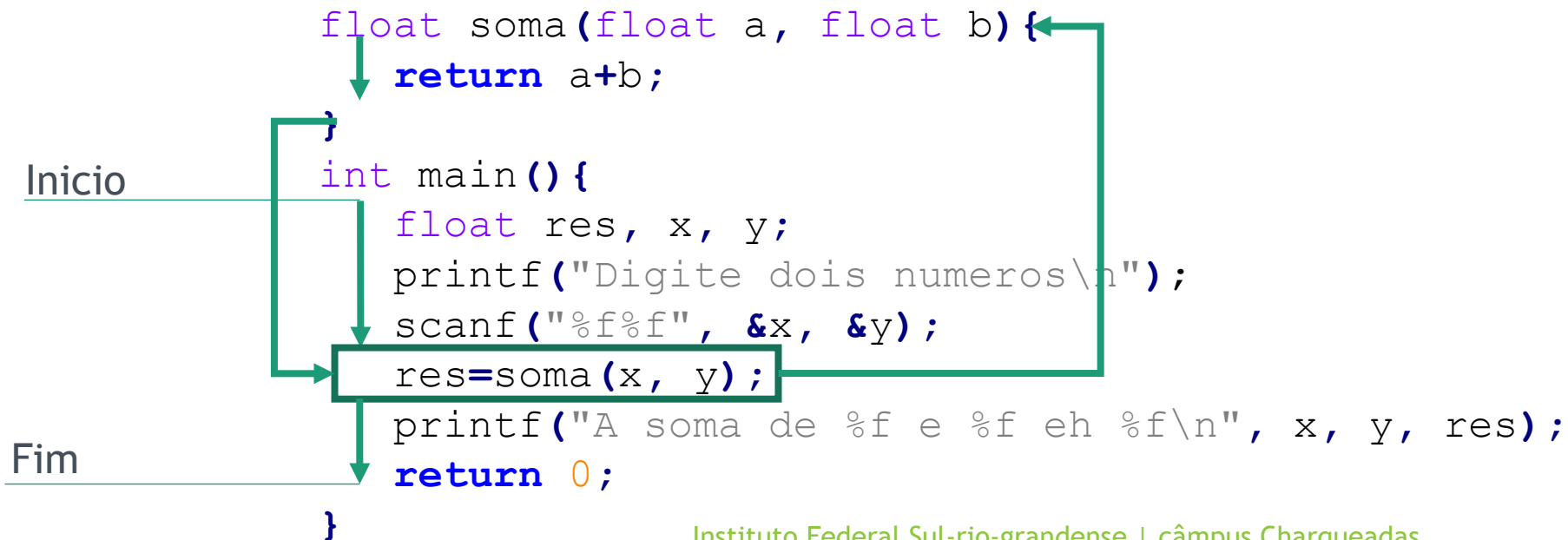
Variavel *res* armazenou o retorno da função soma

Nessa chamada de soma, *a=2* *b=9*

# Funções

## Ordem de Execução

- Ao chamar uma função, o fluxo de execução do programa é pausado até que a função termine a sua execução
  - O fluxo de execução continua sendo sequencial





# Funções

## Escopo de variáveis

- Ao executar este programa, o resultado sera:

```
Global i=10
Antes da funcao i=5
Dentro da funcao i=4
Antes da funcao i=5
```

```
int i=10;
void printDecrementa(int i){
    i=i+1;
    printf("Dentro da funcao i=%i\n", i);
}
int main(){
    printf("Global i=%i\n", i);

    int i=5;

    printf("Antes da funcao i=%i\n", i);
    printDecrementa(i);
    printf("Depois da funcao i=%i\n", i);

    return 0;
}
```

# Exemplos

# Funções - Exemplo 1

Chamando funções corretamente

- Ha diferenças grandes entre as 3 funções

- `leDoisNumsESoma ()`
  - Lista de parametros vazia
  - Retorna vazio
- `printSoma ()`
  - Lista de parametros com 2 float
  - Retorna vazio
- `soma ()`
  - Lista de parametros com 2 float
  - Retorna float

```
void leDoisNumsESoma () {  
    float a, b;  
    printf("Digite dois numeros\n")  
    scanf("%f%f", &a, &b);  
    printf("A soma eh %f\n", a+b);  
}  
void printSoma(float a, float b) {  
    printf("A soma eh %f\n", a+b);  
}  
float soma(float a, float b) {  
    return a+b;  
}  
int main() {  
    float res;  
    leDoisNumsESoma();  
    printSoma(2.0, 2.0);  
    res=soma(3.5, 0.5);  
    printf("A soma eh %f\n", res);  
    return 0;  
}
```

# Funções - Exemplo 1

## Chamando funções corretamente

- A chamada (ver função `main()`) de cada uma das funções necessita atenção
  - `leDoisNumsESoma()`
    - Nada dentro dos parentesis
    - Nada a esquerda do nome da função
  - `printSoma()`
    - Dois numeros float dentro dos parentesis separados por vírgula
    - Nada a esquerda do nome da função
  - `soma()`
    - Dois numeros float dentro dos parentesis separados por vírgula
    - Uma variavel a esquerda do nome da função para receber o retorno float da função

```
void leDoisNumsESoma() {  
    float a, b;  
    printf("Digite dois numeros\n")  
    scanf("%f%f", &a, &b);  
    printf("A soma eh %f\n", a+b);  
}  
  
void printSoma(float a, float b) {  
    printf("A soma eh %f\n", a+b);  
}  
  
float soma(float a, float b) {  
    return a+b;  
}  
  
int main() {  
    float res;  
    leDoisNumsESoma();  
    printSoma(2.0, 2.0);  
    res=soma(3.5, 0.5);  
    printf("A soma eh %f\n", res);  
    return 0;  
}
```

# Funções - Exemplo 1

Chamando funções corretamente

- Mas ao executar este programa, o resultado sera:

```
Digite dois numeros
> 1.5 2.5
A soma eh 4.0
A soma eh 4.0
A soma eh 4.0
```

```
void leDoisNumsESoma () {
    float a, b;
    printf("Digite dois numeros\n")
    scanf("%f%f", &a, &b);
    printf("A soma eh %f\n", a+b);
}

void printSoma(float a, float b){
    printf("A soma eh %f\n", a+b);
}

float soma(float a, float b){
    return a+b;
}

int main(){
    float res;
    leDoisNumsESoma();
    printSoma(2.0, 2.0);
    res=soma(3.5, 0.5);
    printf("A soma eh %f\n", res);
    return 0;
}
```

# Funções - Exemplo 2

Desenvolvendo exercicios com funções

- Desenvolva utilizando 3 passos:
  - 1 - Defina o **cabeçalho** da função
  - 2 - Descreva a **logica** da função
  - 3 - Demonstre a **utilização** da função
- Considere o seguinte enunciado como exemplo

**Somatorio de numeros em um intervalo.** Escreva uma função com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A função deve retornar** a soma de todos os numeros dentro do intervalo dado. Exemplo de `main()`:

- > O somatorio dos numeros entre 1 e 5 eh 15
- > O somatorio dos numeros entre 7 e 9 eh 24

# Funções - Exemplo 2

## 1 - Definindo o cabeçalho

```
int somaIntervalo(int inicio, int fim){  
}
```

Coerencia entre o  
enunciado e o cabeçalho  
da função

**Somatorio de numeros em um intervalo.** Escreva uma função com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A função deve retornar** a soma de todos os numeros dentro do intervalo dado. Exemplo de main():

- > O somatorio dos numeros entre 1 e 5 eh 15
- > O somatorio dos numeros entre 7 e 9 eh 24

# Funções - Exemplo 2

## 2 - Descreva a logica

```
int somaIntervalo(int inicio, int fim){  
    int aux, i, soma=0;  
    if(inicio > fim){  
        aux=inicio;  
        inicio=fim;  
        fim=aux;  
    }  
    for(i=inicio; i<=fim; i++){  
        soma+=i;  
    }  
    return soma;  
}
```

Descreva a logica como voce sempre fez para desenvolver o `main()` e utilize o comando **return** quando necessario

**Somatorio de numeros em um intervalo.** Escreva uma função com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A função deve retornar** a soma de todos os numeros dentro do intervalo dado. Exemplo de `main()`:

- > O somatorio dos numeros entre 1 e 5 eh 15
- > O somatorio dos numeros entre 7 e 9 eh 24



# Funções - Exemplo 2

## 3 - Demonstre o uso

```
int somaIntervalo(int inicio, int fim){
    int aux, i, soma=0;
    if(inicio > fim){
        aux=inicio;
        inicio=fim;
        fim=aux;
    }
    for(i=inicio; i<=fim; i++){
        soma+=i;
    }
    return soma;
}
```

```
int main(){
    int res;
    res = somaIntervalo(1,5);
    printf("O somatorio dos numeros entre 1 e 5 eh %i\n", res);
    printf("O somatorio dos numeros entre 7 e 9 eh ");
    printf("%i\n", somaIntervalo(7,9));
    return 0;
}
```

No `main()` voce escreve um programa para utilizar sua função

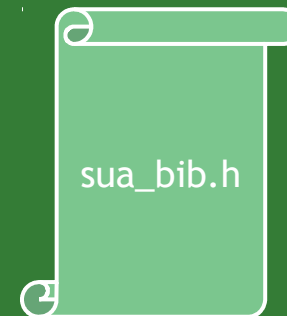
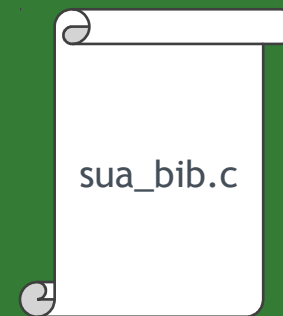
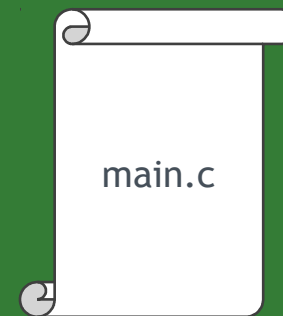
**Somatorio de numeros em um intervalo.** Escreva uma função com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A função deve retornar** a soma de todos os numeros dentro do intervalo dado. Exemplo de `main()`:

- > O somatorio dos numeros entre 1 e 5 eh 15
- > O somatorio dos numeros entre 7 e 9 eh 24

# Programação Estruturada

# Programação Estruturada

## Organização de arquivos



- Em **sua\_bib.h** estão descritos, nesta ordem:
  - Definições de constantes,
  - Todas as variáveis globais,
  - Todas as estruturas definidas pelo programador
    - **struct, enum, union** - ainda serão vistos nesta disciplina
  - Descrição de todos os cabeçalhos das funções descritas em **sua\_bib.c**

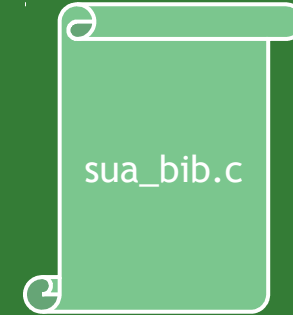
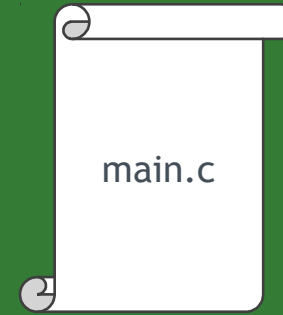
```
#define MAX 5
//TADs - struct, unions, enum

//var globais
int x;

//prototipos das funcoes
int soma(int a, int b);
int multiplica(int a, int b);
int potencia(int a, int b);
```

# Programação Estruturada

## Organização de arquivos



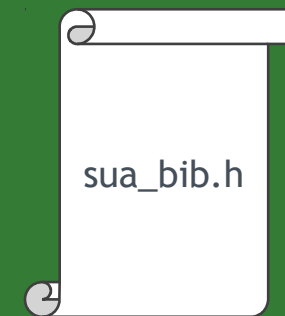
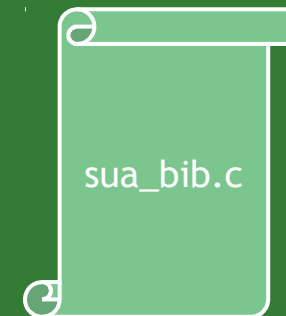
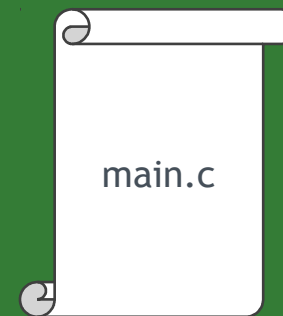
- Em **sua\_bib.c** estão descritos:
  - A lógica de todas as funções declaradas em **sua\_bib.h**

```
#include <stdio.h>
#include "sua_bib.h"
int soma(int a, int b){
    return a+b;
}
int multiplica(int a, int b){
    int i, ac=0;
    for (i = 0; i < b; ++i){
        ac = soma(ac, a);
    }
    return ac;
}
```

```
int potencia(int a, int b){
    int i, ac=1;
    if(b > MAX){
        printf("ERRO - b > 5\n");
        return -1;
    }
    for (i = 0; i < b; ++i){
        ac = multiplica(ac, a);
    }
    return ac;
}
```

# Programação Estruturada

## Organização de arquivos



- Em destaque o **REUSO** das funções

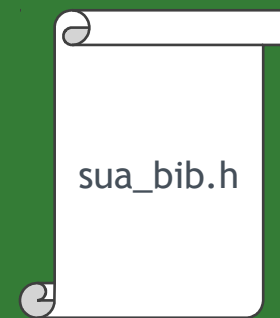
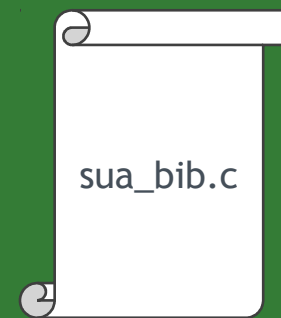
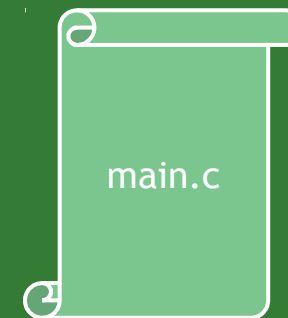
- soma () dentro da função multiplica ()
- multiplica () dentro da função potencia ()

```
#include <stdio.h>
#include "sua_bib.h"
int soma(int a, int b){
    return a+b;
}
int multiplica(int a, int b){
    int i, ac=0;
    for (i = 0; i < b; ++i){
        ac = soma(ac, a);
    }
    return ac;
}
```

```
int potencia(int a, int b){
    int i, ac=1;
    if(b > MAX){
        printf("ERRO - b > 5\n");
        return -1;
    }
    for (i = 0; i < b; ++i){
        ac = multiplica(ac, a);
    }
    return ac;
}
```

# Programação Estruturada

## Organização de arquivos



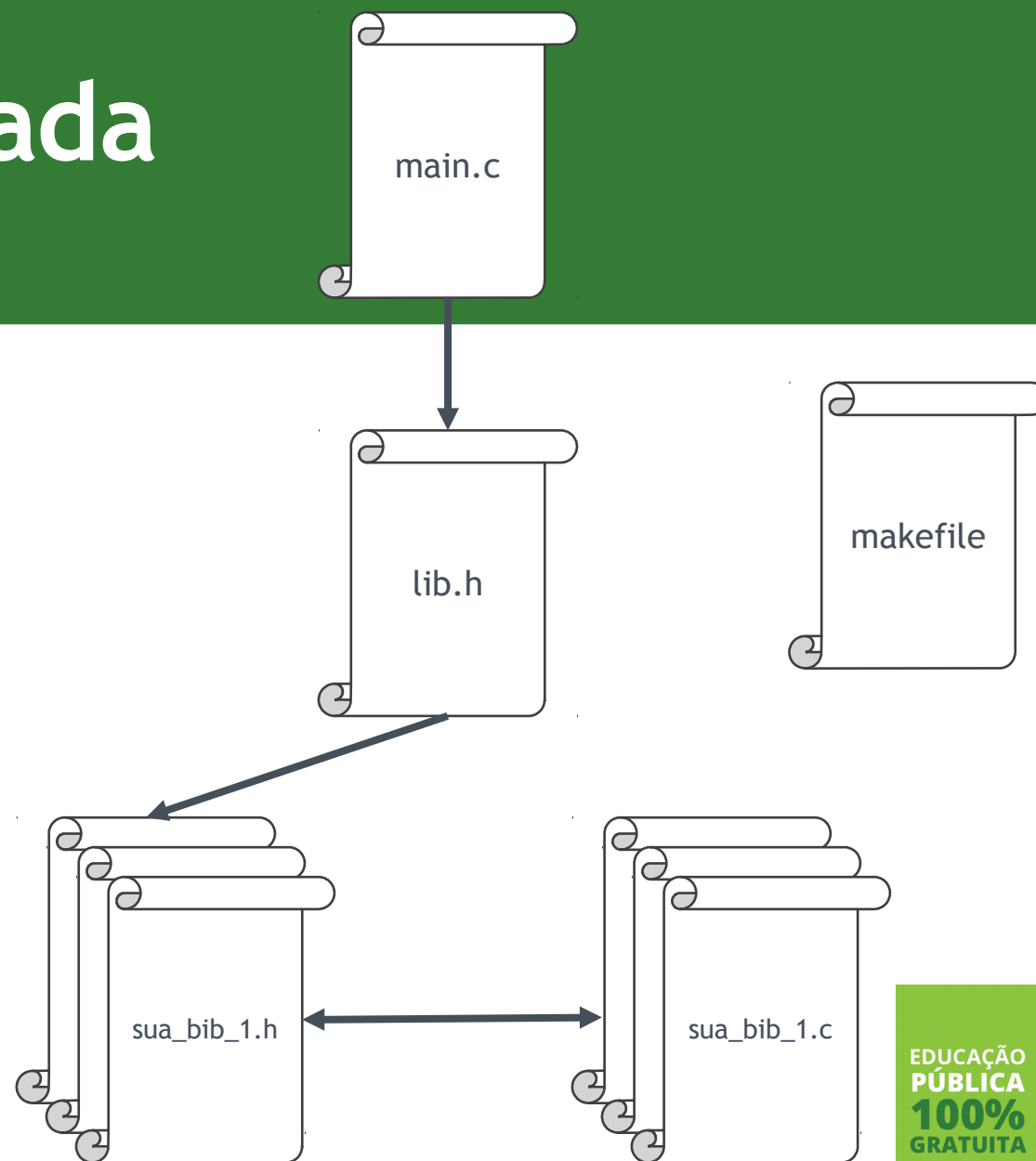
- Em **main.c** esta descrito:
  - Apenas a função `main()`
  - E necessario fazer um include de sua biblioteca

```
#include <stdio.h>
#include "sua_bib.h"
int main () {
    int x=4, y=2;
    printf("### DEMOS ###\n");
    printf("soma = %i\n", soma(x, y));
    printf("multiplica = %i\n", multiplica(x, y));
    printf("potencia = %i\n", potencia(x, y));
    return 0;
}
```

# Programação Estruturada

Organização de arquivos com mais de uma biblioteca

- Para incluir mais de uma biblioteca propria
  - Apenas a função `lib.h` e incluída em `main.c`
  - Todos os includes de `sua_bib_N.h` devem ser feitos em `lib.h`
  - O arquivo `makefile` contém o script para montar o programa descrito em `main.c`
- Esse é o formato que entregarei os códigos a partir de agora
- **Codeblocks, DevC++, VisualStudio** também realizam a montagem de código estruturado por meio de **Projetos**



# Projeto no Codeblocks



# Projeto no Codeblocks

## Demonstração

- Demonstração de criação de um projeto estruturado nas ferramentas disponíveis no campus
- **Nao entendeu a DEMO? consulte os tutoriais**
  - [Codeblocks](#)
  - [DevC++](#)
  - [VisualStudio](#) (para C++, mas e igual em C)

MUITO  
**OBRIGADO**

Prof. André del Mestre

[www.ifsul.edu.br](http://www.ifsul.edu.br)  
[almmartins@charqueadas.ifsul.edu.br](mailto:almmartins@charqueadas.ifsul.edu.br)