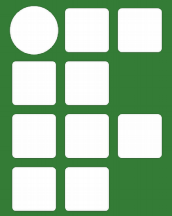


INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Charqueadas

EDUCAÇÃO PÚBLICA 100% GRATUITA

Funçoes em C

Programaçao Estruturada

Prof. André del Mestre

Principais Conceitos

Funçoes

Forma Geral

- `tipo_de_retorno` - valor que a funçao pode retornar
  - Funçao sem retorno: void
  - Funçao com um retorno, escolha: float, char, int, ...
- `nome_da_funcao` - segue as mesmas regras de declaraçao de variaveis
  - Iniciar com letra, nao pode usar caracteres especiais, palavras-chave do sistema, etc...
- `lista_de_parametros` - lista de valores passados para funçao processar
  - Pode-se definir quantos parametros quiser
- `secao de comandos` - descriçao da logica da funçao
  - Use operadores logicos, aritmeticos, chamadas de funçoes, estruturas de seleçao e repetiçao

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

Funçoes

Lista de Parametros

- `lista_de_parametros` - Forma Geral
  - `tipo1 nome1, tipo2 nome2, ..., tipoN nomeN`

  - Os `nomes` dos parametros seguem as mesmas regras de declaraçao 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;
}
```

Funçoes

Retorno de Funçoes

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

```c
//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!
}
```

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

Funçoes

Comando return

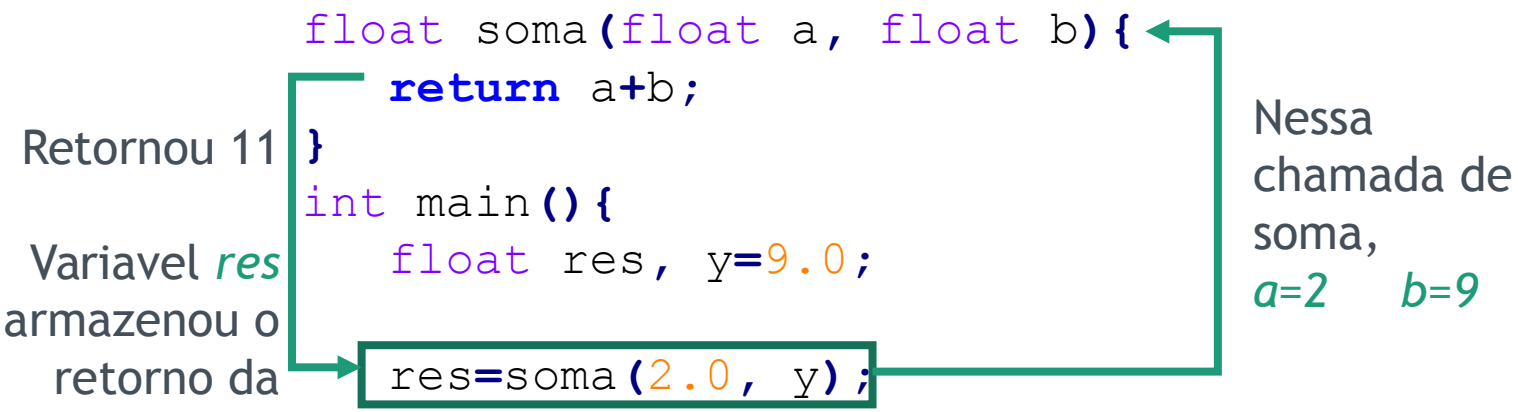
- O valor gerado pela funçao e dado pelo comando **return**
  - **return** valor, variavel ou expressao;
- O **return** encerra a execuçao de qualquer funçao
- Preocupe-se com a compatibilidade de tipos ao retornar funçoes

```
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");
}
```

Funçoes

Passagem de parametro

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

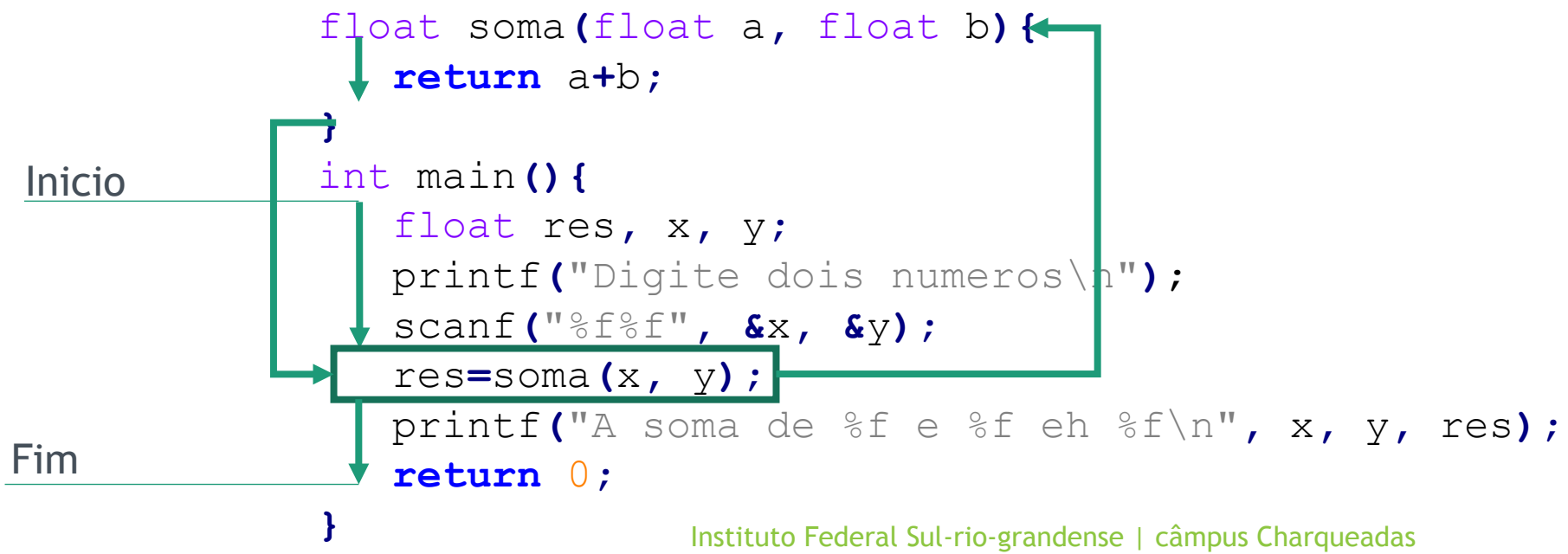



```c
float soma(float a, float b){
    return a+b;
}
int main(){
    float res, y=9.0;

    res=soma(2.0, y);

    printf("A soma eh %f\n", res);
    return 0;
}
```

Funçoes

Ordem de Execuçao

- Ao chamar uma funçao, o fluxo de execuçao do programa e pausado ate que a funçao termine a sua execuçao
  - O fluxo de execuçao continua sendo sequencial



Funçoes

Escopo de variaveis

- Ao executar este programa, o resultado sera:

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

```c
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

Funçoes - Exemplo 1

Chamando funçoes 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

```c
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;
}
```

Funçoes - Exemplo 1

Chamando funçoes 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;
}
```

Funçoes - Exemplo 1

Chamando funçoes 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
```

```c
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;
}
```

Funçoes - Exemplo 2

Desenvolvendo exercicios com funçoes

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

Somatorio de numeros em um intervalo. Escreva uma funçao com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A funçao 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
```

Funçoes - Exemplo 2

1 - Definindo o cabeçalho

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

}
```

Coerencia entre o enunciado e o cabeçalho da funçao

Somatorio de numeros em um intervalo. Escreva uma funçao com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A funçao 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
```

Funçoes - 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 funçao com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A funçao 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
```

Funçoes - Exemplo 2

3 - Demonstre o uso

```c
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;
}
```

```c
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 funçao

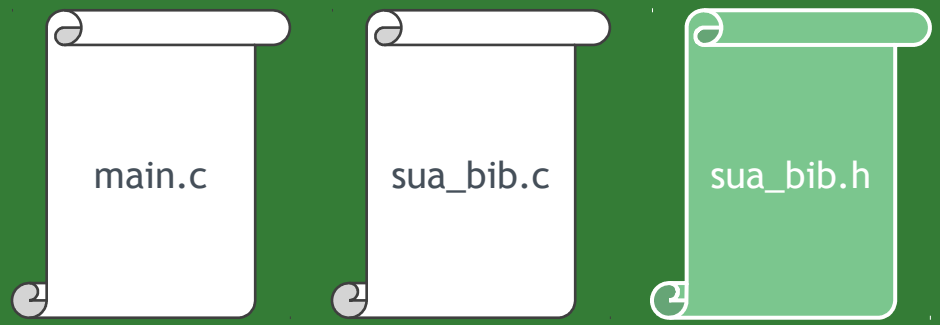
Somatorio de numeros em um intervalo. Escreva uma funçao com **dois parametros**: um numero inteiro para o inicio do intervalo e um numero inteiro para o fim do intervalo. **A funçao deve retornar** a soma de todos os numeros dentro do intervalo dado. Exemplo de `main()`:

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

Programaçao Estruturada

Programaçao Estruturada

Organização de arquivos

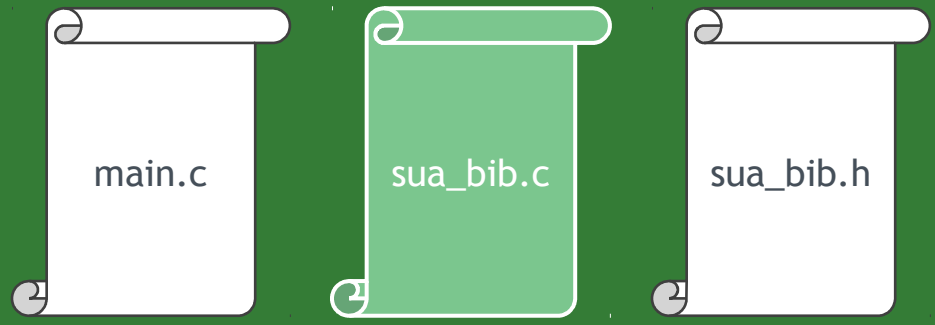


- Em **sua_bib.h** estao descritos, nesta ordem:
  - Definiçoes de constantes,
  - Todas as variaveis globais,
  - Todas as estruturas definidas pelo programador
    - struct, enum, union – ainda serao vistos nesta disciplina
  - Descriçao de todos os cabeçalhos das funçoes descritas em **sua_bib.c**

```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);
```

Programaçao Estruturada

Organização de arquivos



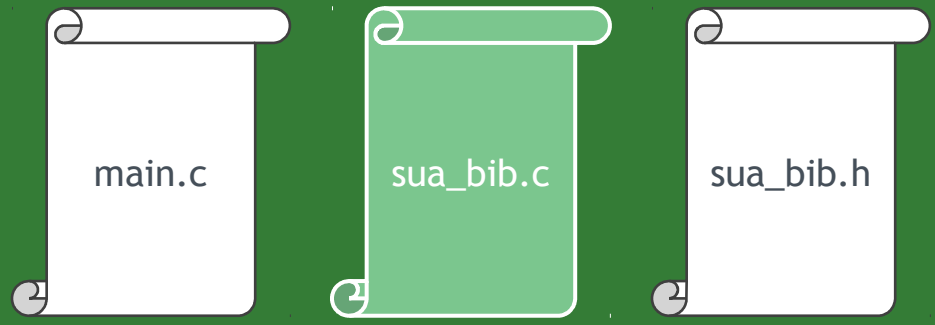
- Em **sua_bib.c** estao descritos:
  - A logica de todas as funçoes declaradas em **sua_bib.h**

```c
#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;
}
```

```c
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;
}
```

Programaçao Estruturada

Organização de arquivos



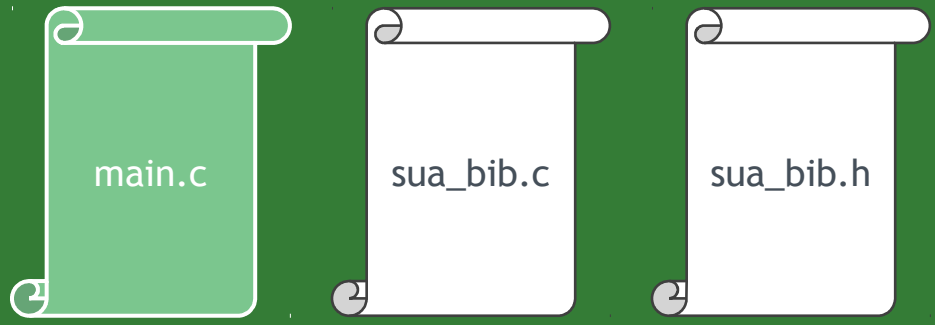
- Em destaque o **REUSO** das funções
  - `soma()` dentro da função `multiplica()`
  - `multiplica()` **dentro da função** `potencia()`

```c
#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;
}
```

```c
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;
}
```

Programaçao Estruturada

Organização de arquivos



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

```c
#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;
}
```

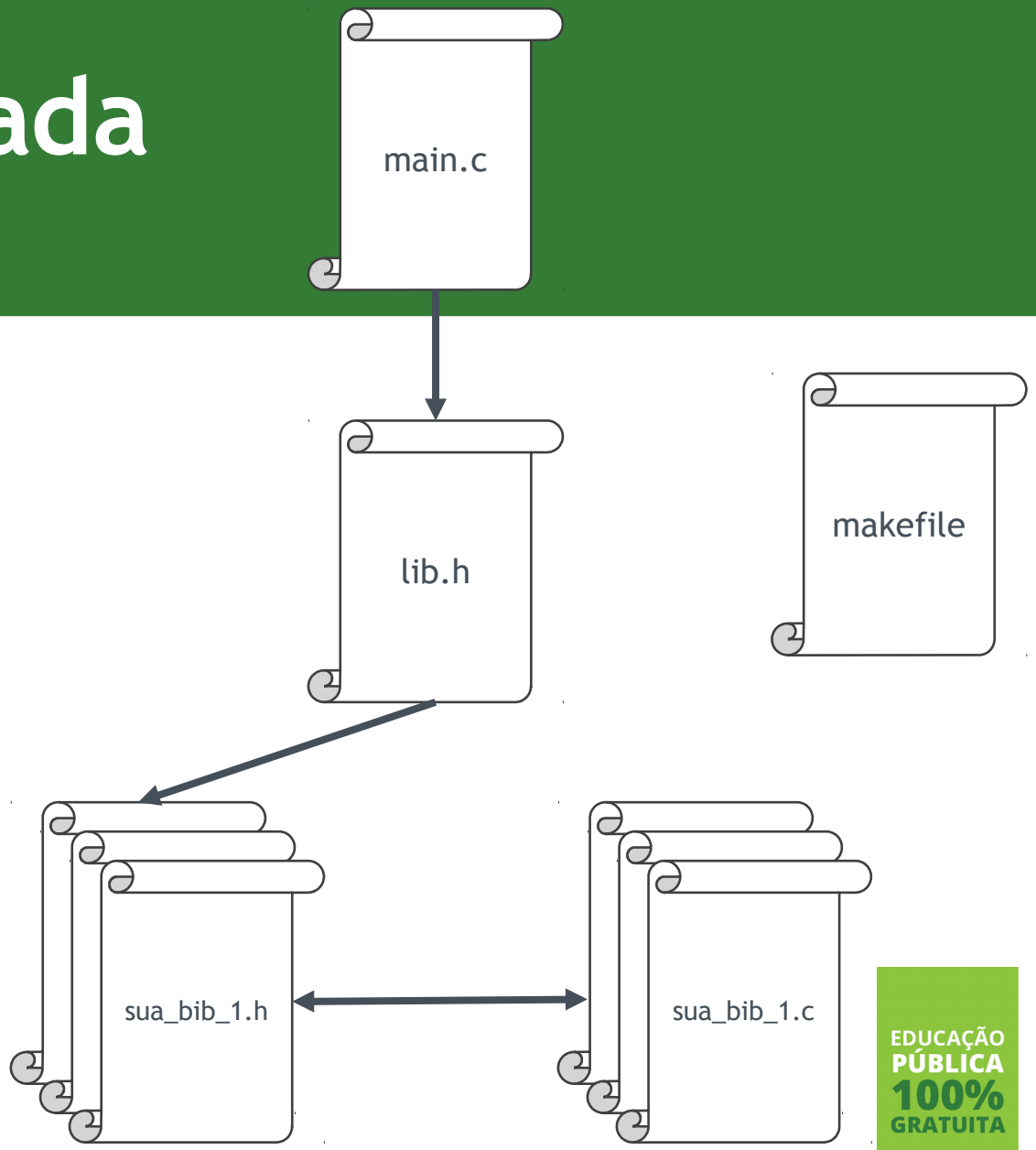
Programaçao Estruturada

Organização de arquivos com mais de uma biblioteca

- Para incluir mais de uma biblioteca propria
  - Apenas a funçao `lib.h` e incluida em `main.c`
  - Todos os includes de `sua_bib_N.h` devem ser feitos em `lib.h`
  - O arquivo `makefile` contem o script para montar o programa descrito em `main.c`

  - Esse e o formato que entregarei os codigos a partir de agora

  - **Codeblocks, DevC++, VisualStudio** tambem realizam a montagem de código estruturado por meio de **Projetos**



Projeto no Codeblocks

Projeto no Codeblocks

Demonstraçao

- Demonstraçao de criaçao de um projeto estruturado nas ferramentas disponiveis 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
almmartins@charqueadas.ifsul.edu.br