Computação eletrônica: Funções

Gurvan Huiban
ghuiban@cin.ufpe.br

13 de maio de 2014

Plano de aula

Introdução

O que é?

Bloco de instruções desenvolvido para executar alguma atividade específica.

Porque?

- Quebrar o programa em vários “blocos”
- Cada faz só uma tarfa

É novo?

- o `main` é uma função!
- `printf`, `scanf`, `sqrt`...
- Agora: criação de funções

Função

Definição

- Bloco de instruções
- Execução de alguma atividade específica
- Chamada durante a execução do programa
- Uma função pode chamar outras funções

Observação

- Uma função pode ser chamada a partir de outras funções.
- Estrutura semelhante à da função `main()`;
- A função `main()`:
  - Nome especial
  - Primeira a ser chamada (automaticamente) na execução do programa

Exemplo: definição da função linha

Função linha

```
// Funcao que escreve 20 '_' consecutivos
void linha(void)
{
    int i;
    for(i=0; i<20; ++i)
    {
        printf("_");
    }
    printf("\n");
}
```

Exemplo: chamada da função linha

Função main

```
int main(void)
{
    linha(); // Chamando a funcao linha
    printf("Titulo\n");
    linha(); // Chamando a funcao linha
}
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

_____

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

_____

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
____________________
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
____________________
```

Linha: Passo a passo

```
// Funcao que escreve 20 '_' a seguir
void linha(void)
{
   int i;
   for(i=0; i<20; ++i)
   {
      printf("_");
   }
   printf("\n");
}

int main(void)
{
   linha(); // Chamando a funcao linha
   printf("Titulo\n");
   linha(); // Chamando a funcao linha
}
```

Tela

```
____________________
Titulo
____________________
```

Modularização do código

Usar funções:

- evita que se escreva o mesmo código inúmeras vezes em um mesmo programa.
- economiza memória.
- é mais fácil manter (uma função, uma tarefa simples).
- permite a construção de bibliotecas.

Chamada de função

- Use o nome da função como se fosse um comando

```
linha();
```

- Argumentos podem ser necessários.
- Uma função pode retornar um resultado.:

```
res = soma(12,15);
```

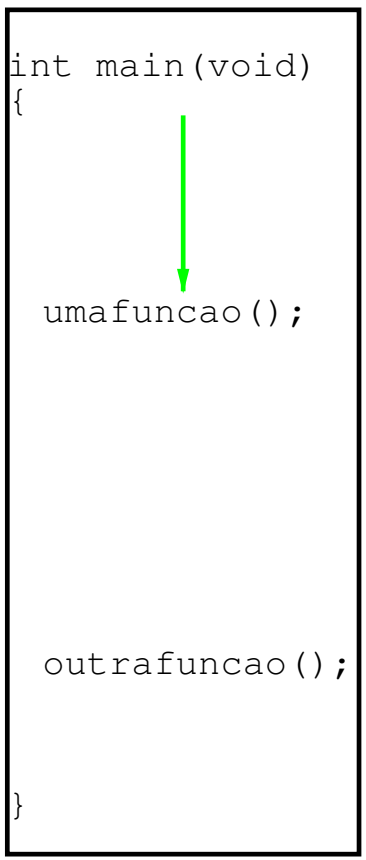
Chamada de função

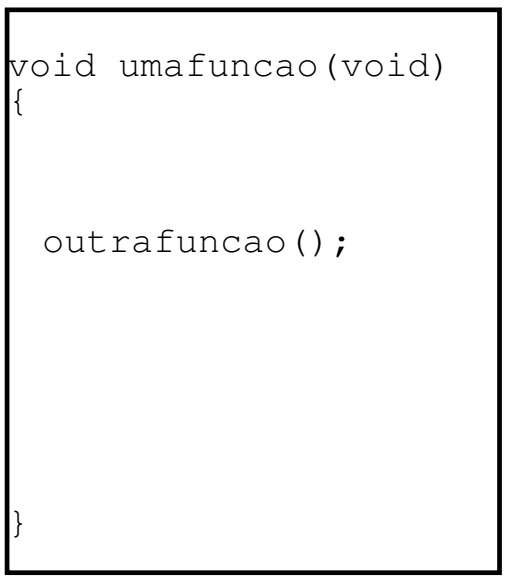
```
int main(void)
{


  umafuncao();




  outrafuncao();


}
```

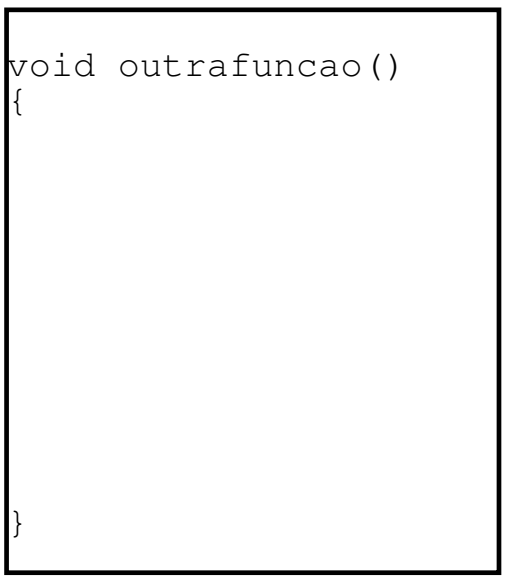
```
void umafuncao(void)
{


  outrafuncao();




}
```

```
void outrafuncao()
{




}
```

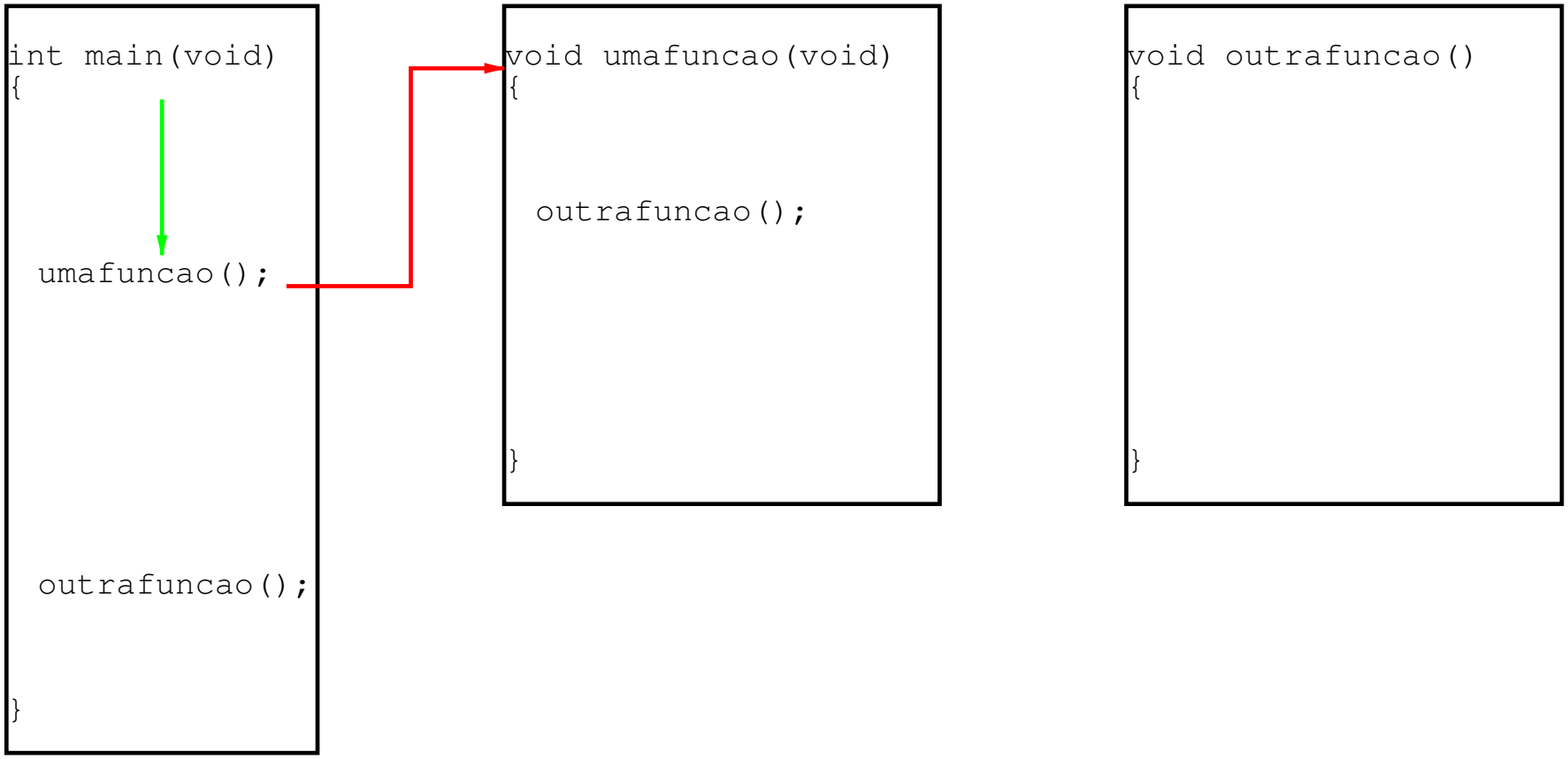
Chamada de função



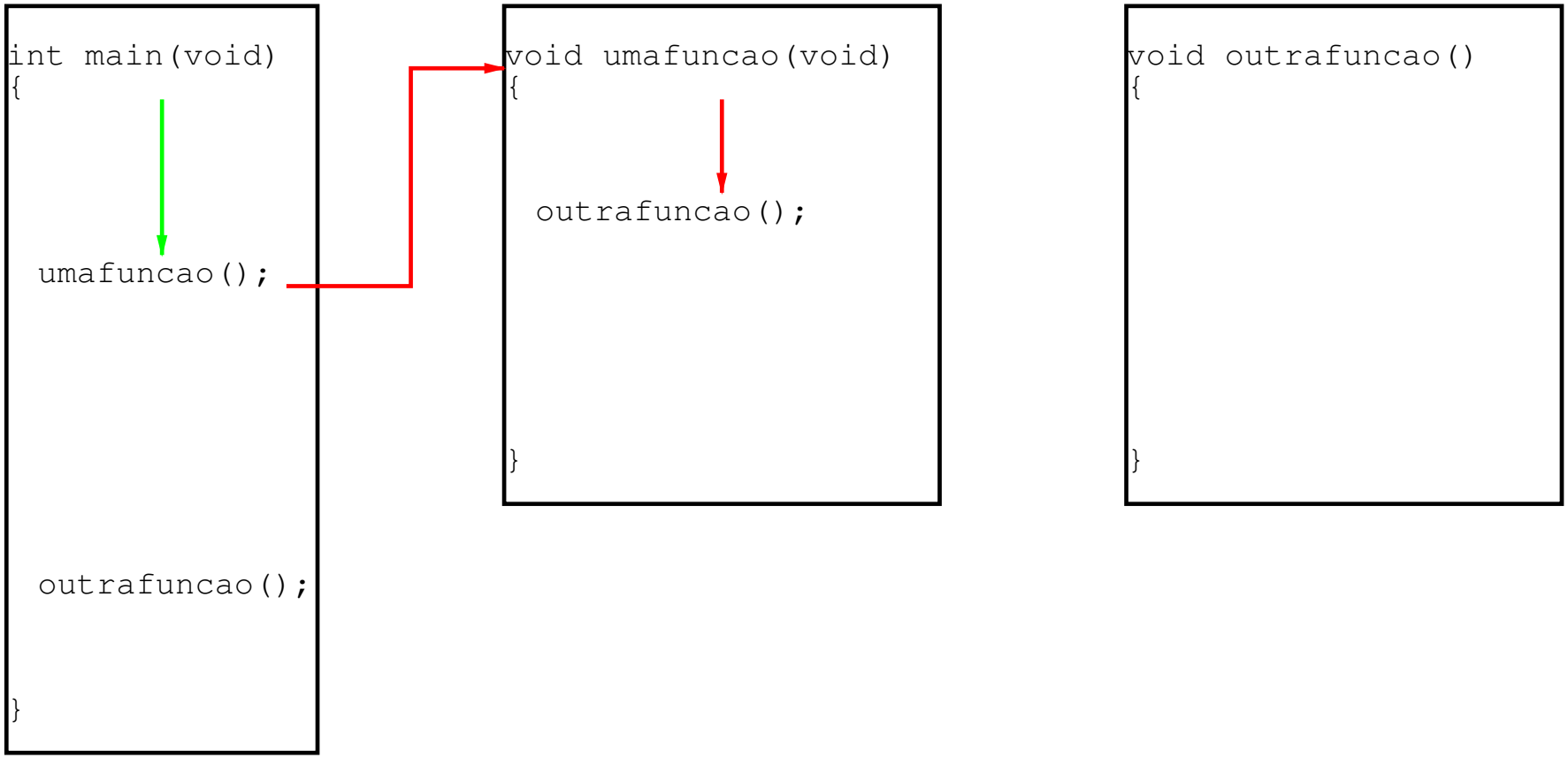




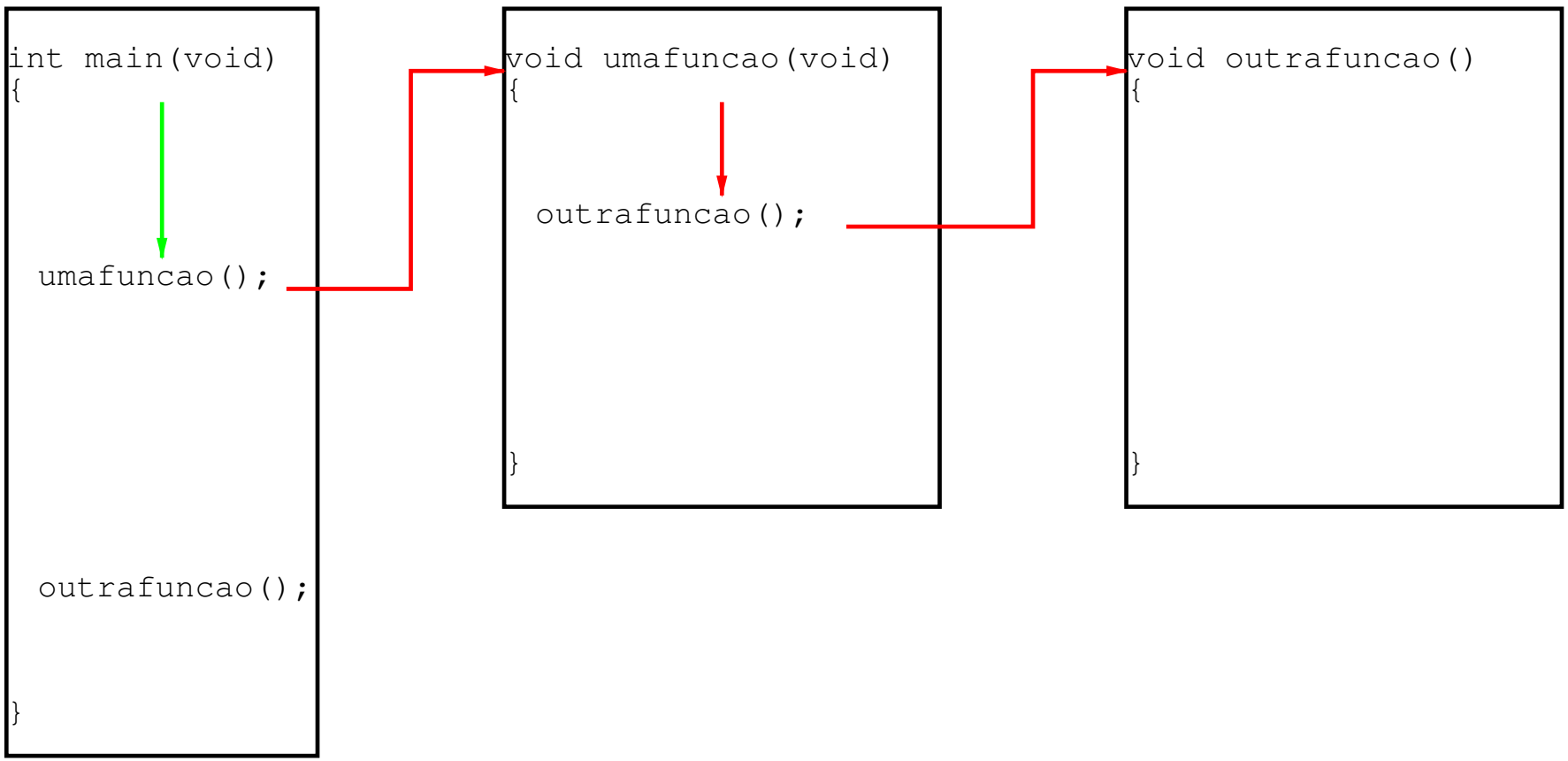
Chamada de função



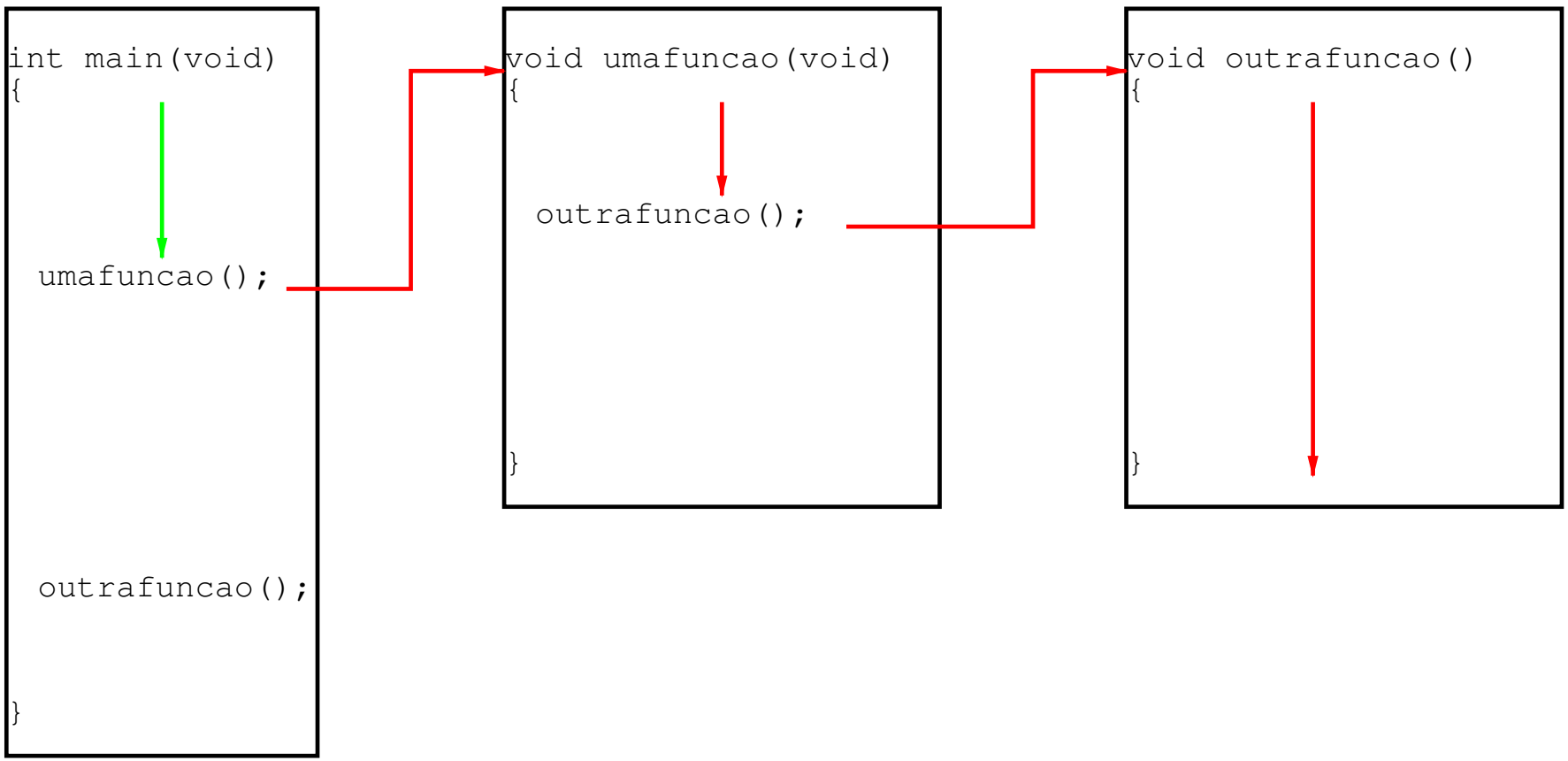
Chamada de função



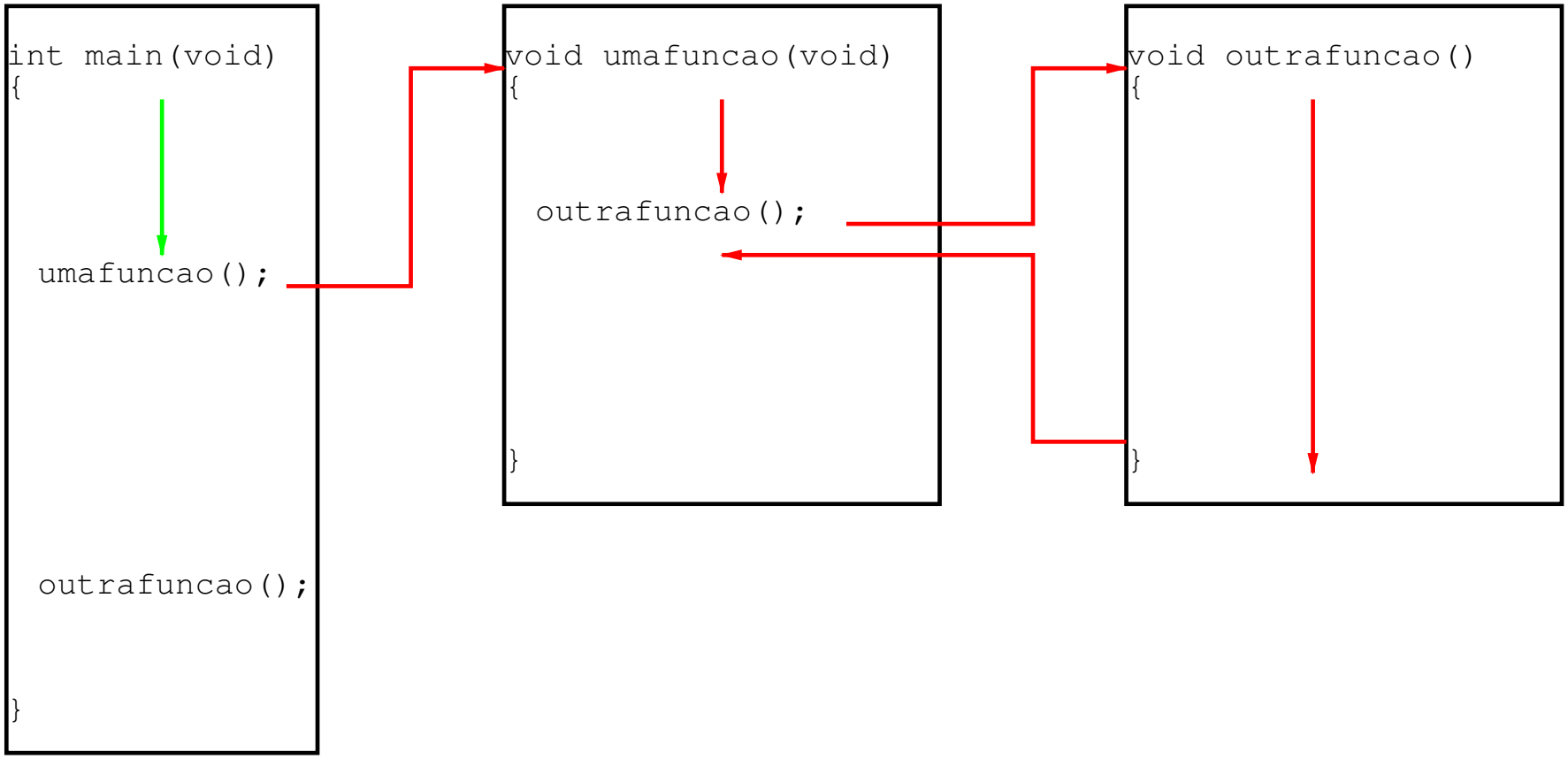
Chamada de função



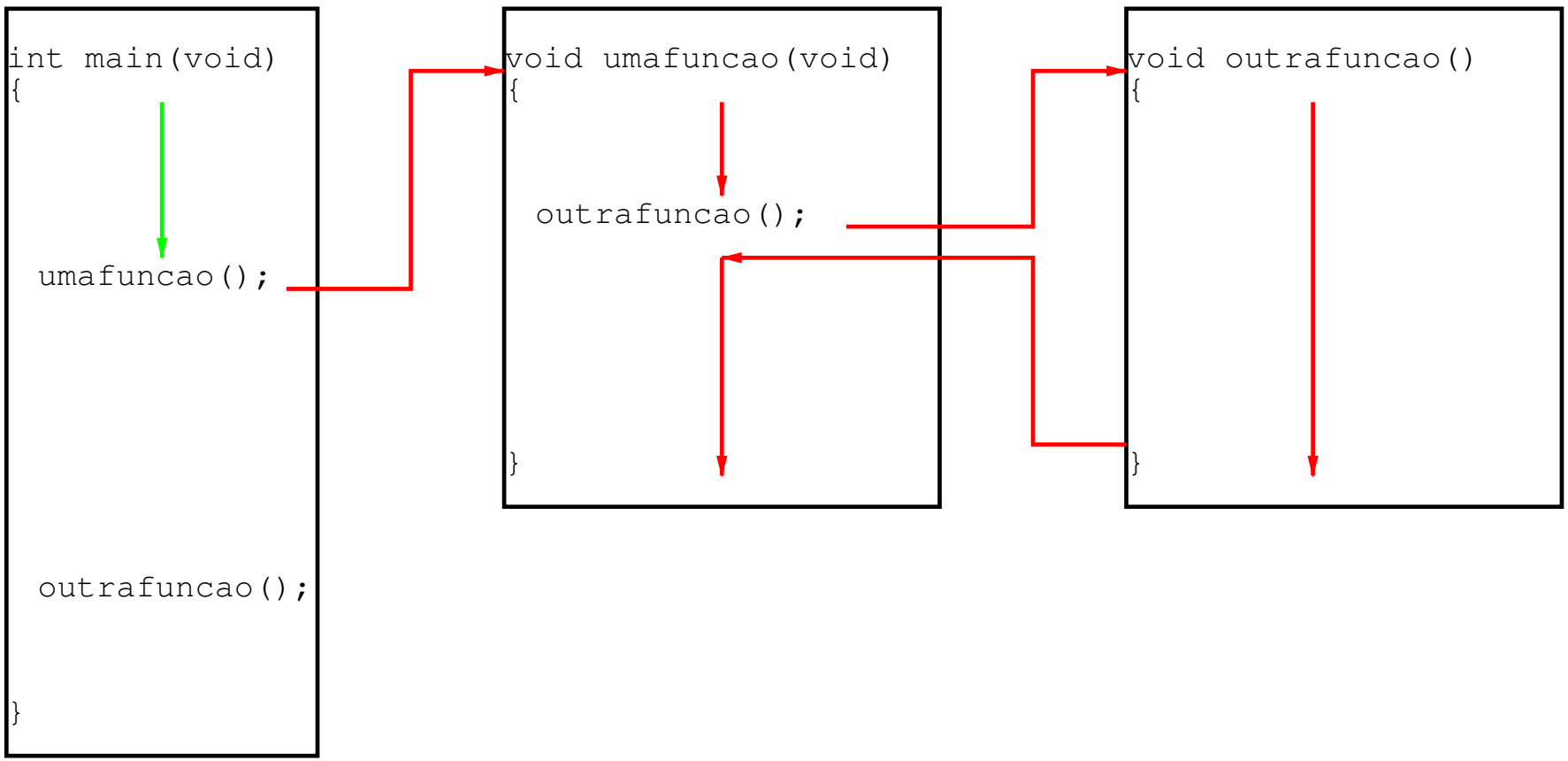
Chamada de função



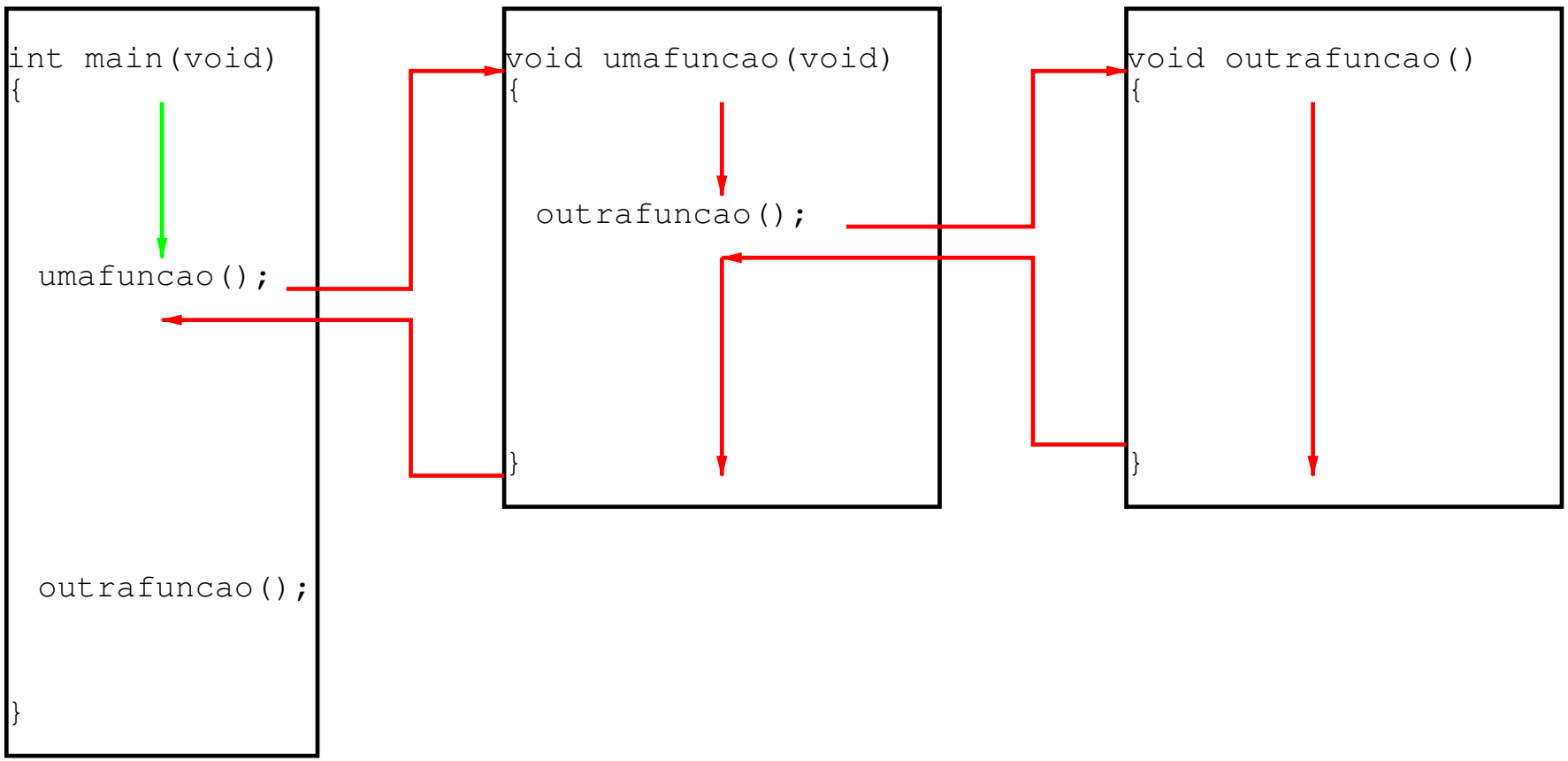
Chamada de função



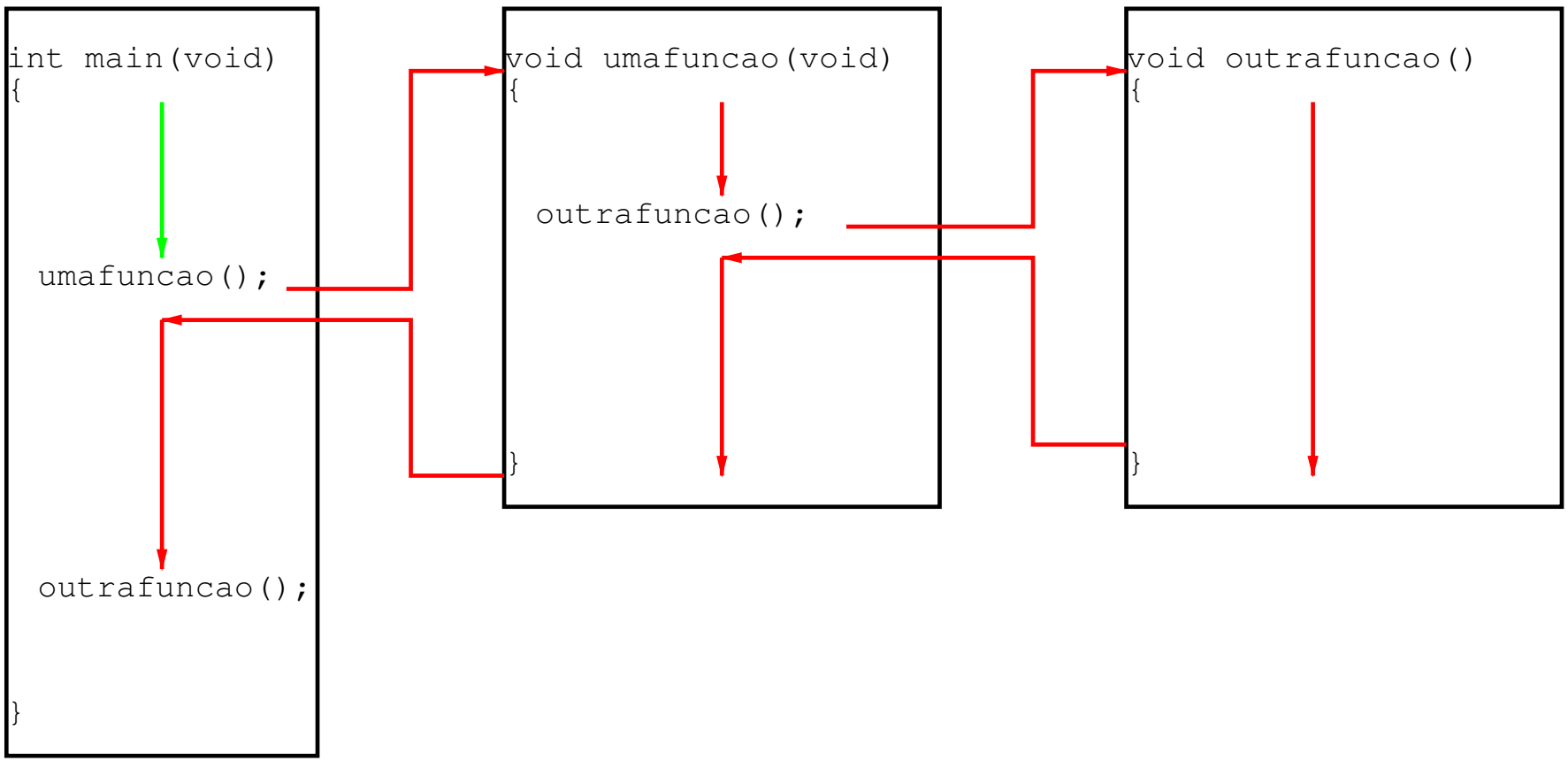
Chamada de função



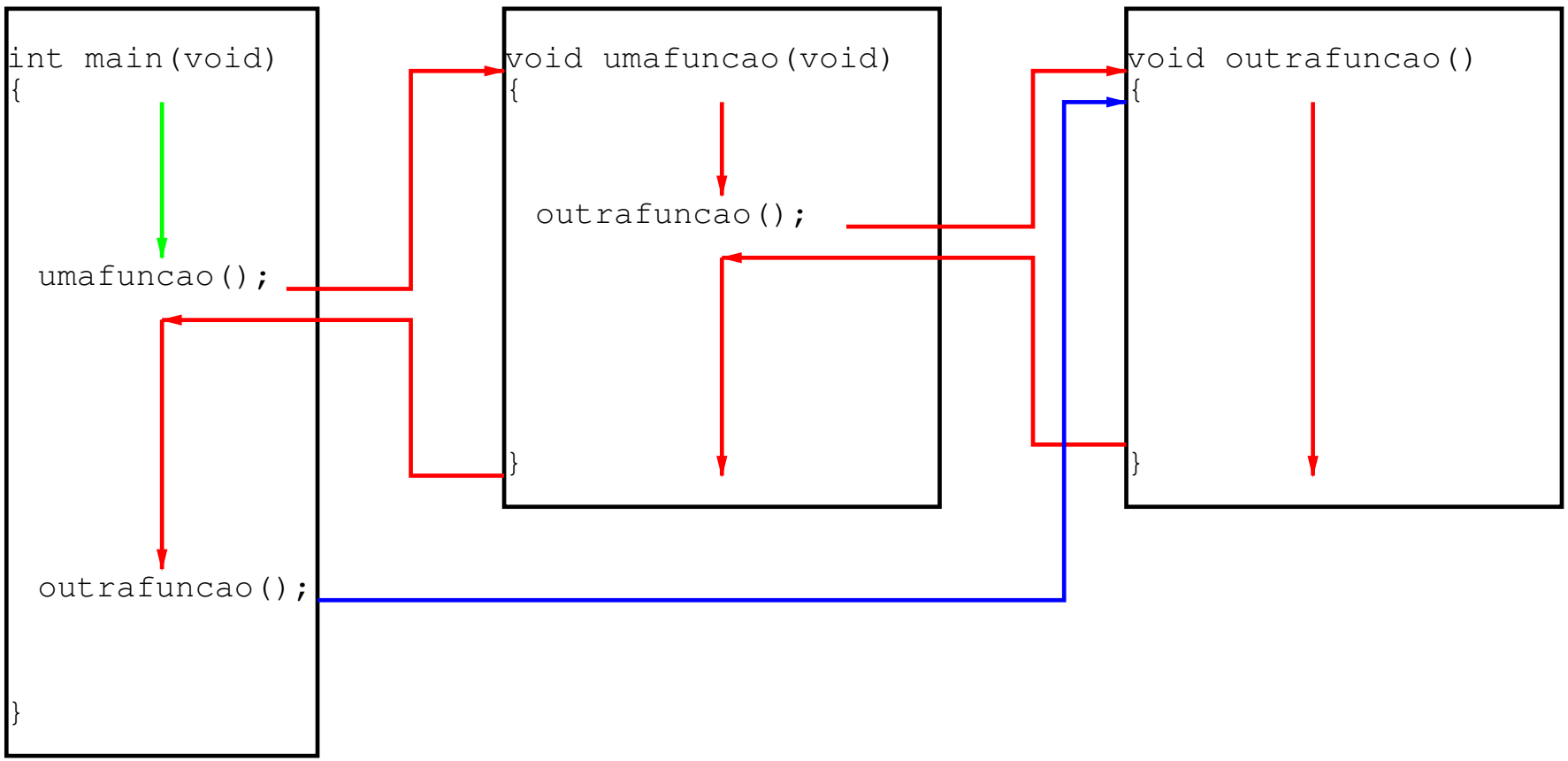
Chamada de função



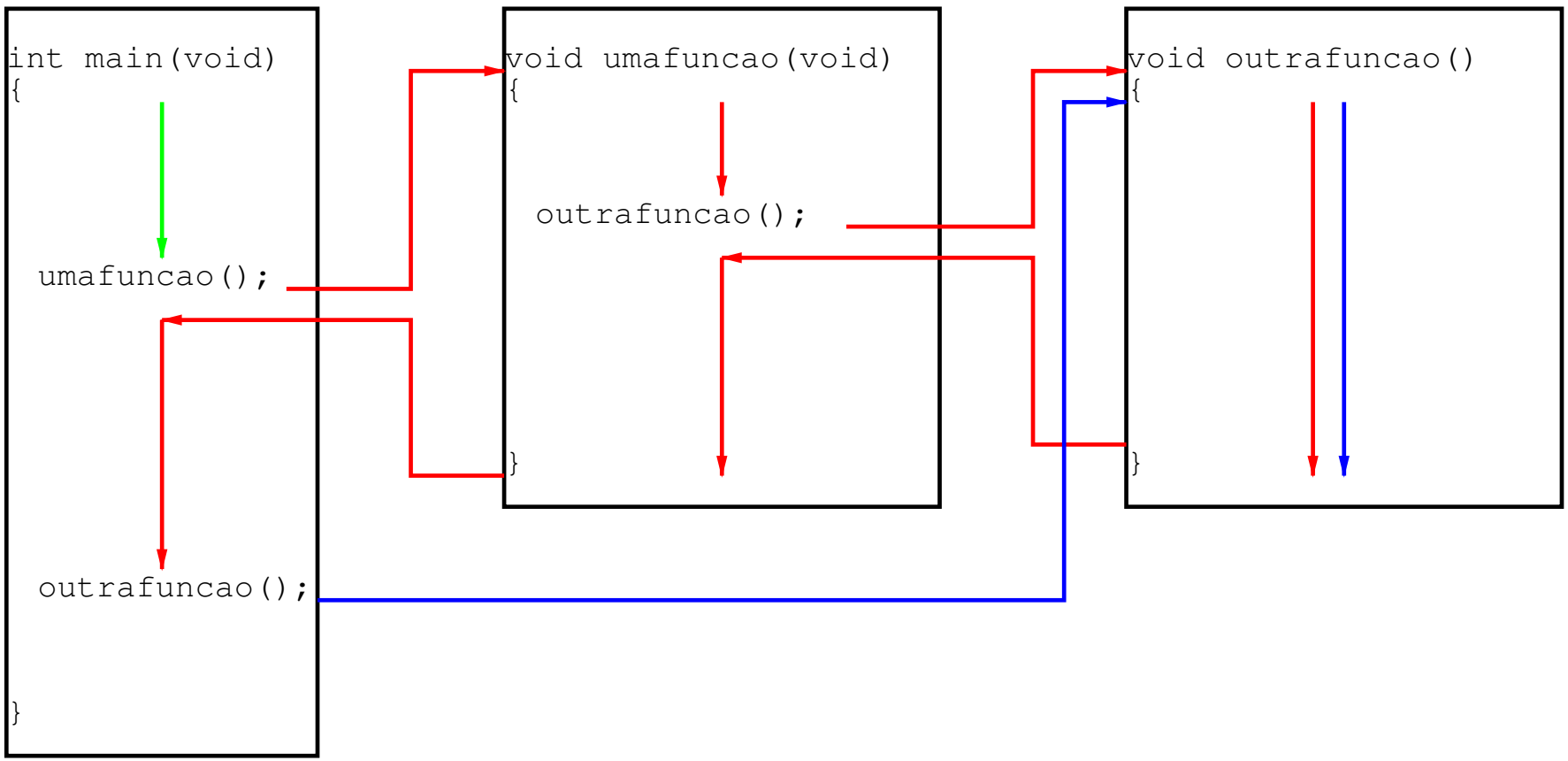
Chamada de função



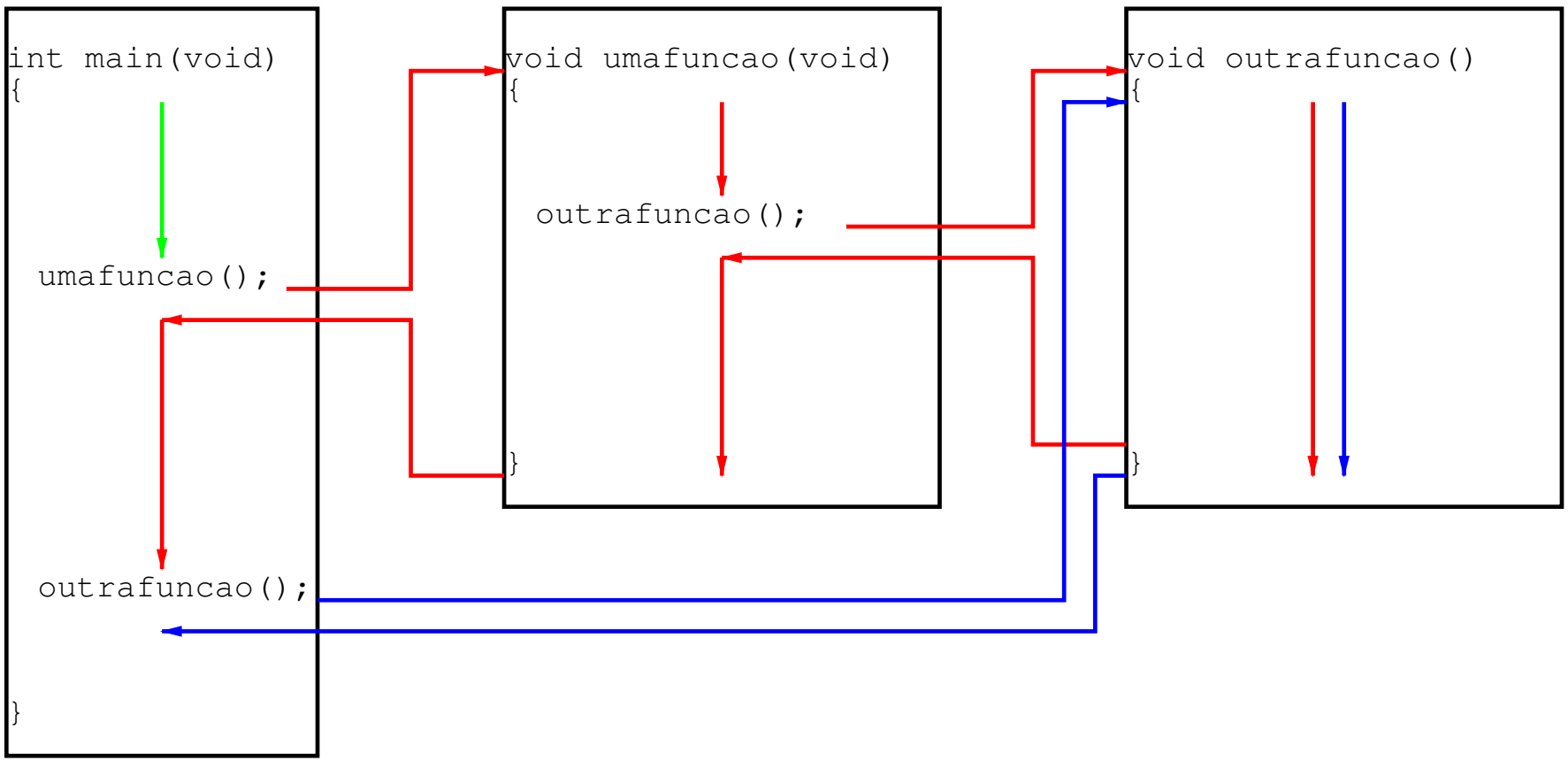
Chamada de função



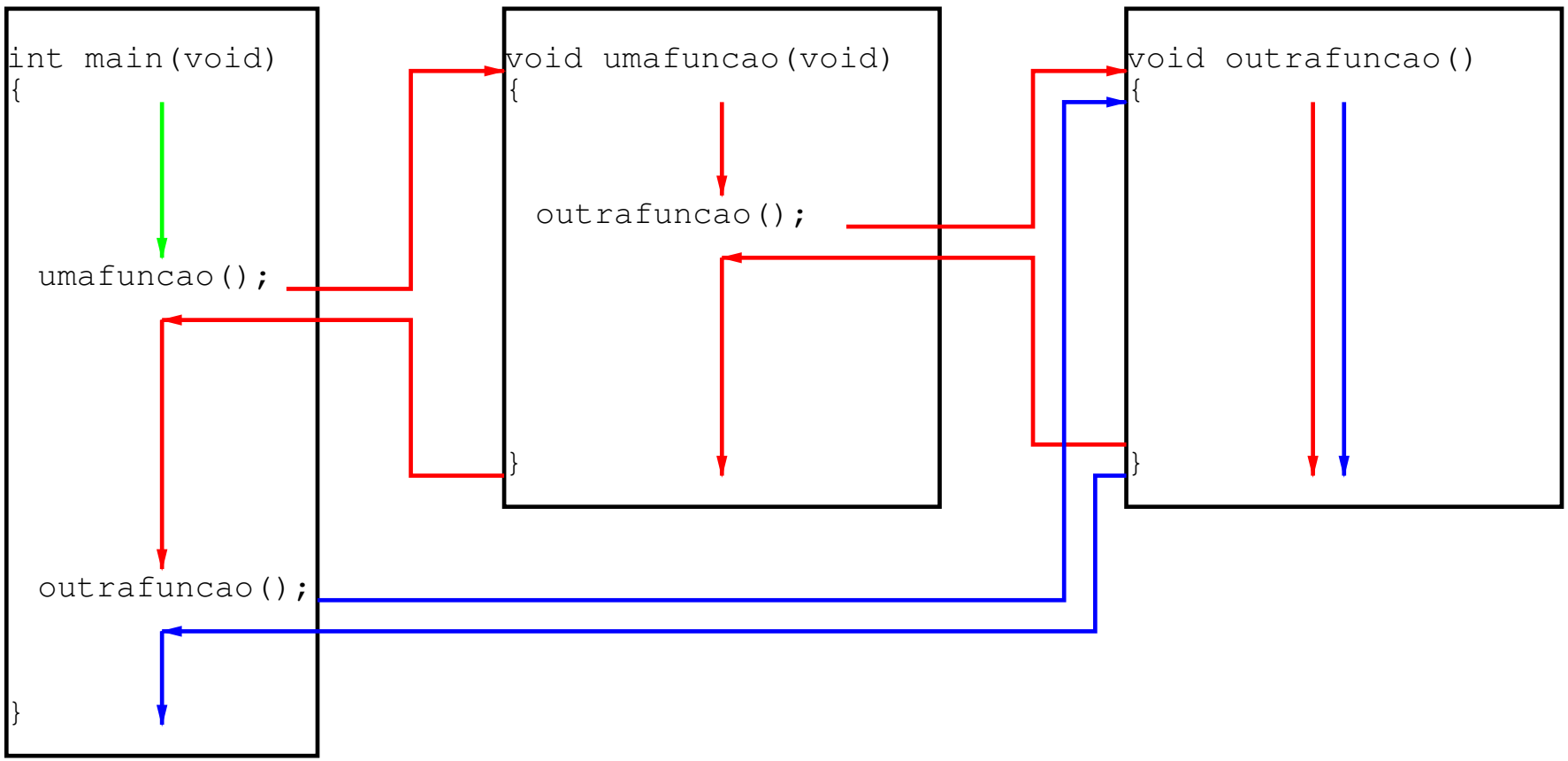
Chamada de função



Chamada de função



Chamada de função



Obervações

Chamadas de função

- Que define os argumentos e o retorno é o protótipo.
- Os parênteses que seguem as funções servem para diferenciar uma chamada de função de uma variável.
- Para algumas funções é necessário fornecer argumentos:
  - valor fixo: `sqrt(5);`
  - variável: `sqrt(val);`
  - resultado de outra função: `sqrt(sqrt(5));`
- Os argumentos são separados por ’,’

Estrutura do arquivo

- Uma função deve ser declarada fora de outras funções;
- As funções devem ser conhecidas pelo compilador no momento de sua chamada:
  ⇒Declarar as funções antes das funções que as chamam.

Estrutura do arquivo

1. `includes`
2. `variáveis globais`
3. `função1`
4. `função2`

   `...`
5. `função main`

Protótipos

```
tipo_retorno nome_funcao(parametros)
```

é o protótipo da função. Exemplos:

- `void linha(void)`
- `int lerNumeroPositivo(void)`
- `void escreveTabuada(int valor)`
- `float distancia(float xa, float ya, float xb, float yb)`

Sintaxe

Declaração

```
tipo_retorno nome_funcao(parametros)
{
  Declaracao de variaveis
  Sequencia de instrucoes
}
```

- `tipo_retorno`: tipo do resultado (`int`, `char`, ...)
  `void` se não retorna resultado
- `nome_funcao`: Identificador da função
- `parâmetros`: Parâmetros de chamada (usar `void` se vazio)
- `Declaração de variáveis`: Variáveis locais da função
- `Sequência de instruções`: Instruções a serem executadas quando a função for chamada

Comando `return`

Retorno de função

- Usar o comando:
  `return (valor_a_retornar);`
- `return` encerra a execução da função.
- Instruções após o comando `return` não serão executadas.

Exemplo

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

Comando `return`

Cuidado

A função seguinte vai retornar 0

```
int inicializacao(void)
{
    int a = 0;
    return (a);
    a = 1;
    return (a);
}
```

Exercício

Função linha:

Escrever uma função que imprima uma linha de 20 caracteres '*' na tela:

```
void linha(void)
```

Função lerNumeroPositivo

Escrever uma função que fique pedindo ao usuário um número inteiro. Se o número for positivo finalizar e retornar o número.

```
int lerNumeroPositivo(void)
```

Leitura e impressão de um número positivo

Escrever um programa que imprima um número positivo inteiro fornecido pelo usuário entre duas linhas de 20 caracteres '*'.

Parâmetros

Definição

- Conjunto de variáveis utilizadas para a função receber os valores para os quais a função deve ser executada.
- Estes valores são chamados argumentos, que constituem os dados de entrada da função.
- Na declaração da função, devemos declarar os parâmetros e o tipo deles.

Exemplo

Para escrever uma tabuada de um inteiro qualquer, a função deve saber para qual inteiro imprimir a tabuada.

```
void escreveTabuada(int num)
```

Parâmetros

Sintaxe

```
tipo nomeProc(tipo1 var1, tipo2 var2 ...);
```

Observação

- Uma função pode ter mais que um parâmetro.
- Os tipos dos parâmetros podem ser diferentes.
- Na função, acesse os valores dos parâmetros com o identificador dele:

  ```
  printf("%d\n",var1);
  ```

- A ordem dos argumentos é a mesma que a ordem dos parâmetros.

Exercício

Distância entre dois pontos

Escrever a função que retorna a distância entre os pontos de coordenados `(xa,ya)` e `(xb,yb)`:

```
float distancia(float xa, float ya,
                float xb, float yb)
```

Escrever um programa em C que peça ao usuário dois pontos $A = (x_a, y_a)$ e $B = (x_b, y_b)$ e que imprima na tela a distância entre A e B.

Definição de protótipos

Para definir o protótipo de uma função, temos que responder às seguintes perguntas:

- A função retorna algum valor?
- De qual tipo?
- A função precisa de quais dados?
- Quais são os tipos deles?

Protótipo

```
tipo_retorno nome_funcao(tipo1 par1, tipo2 par2)
```

Definição de protótipos

Para definir o protótipo de uma função, temos que responder às seguintes perguntas:

- A função retorna algum valor?
- De qual tipo?
- A função precisa de quais dados?
- Quais são os tipos deles?

Protótipo

```
tipo_retorno nome_funcao(tipo1 par1, tipo2 par2)
```

Escope de variáveis

Definição

- O programa é divido em blocos
- Quais variáveis podemos acessar de onde?
- Isso é o escope das variáveis

Tipos de variáveis

- Variáveis locais
- Variáveis globais

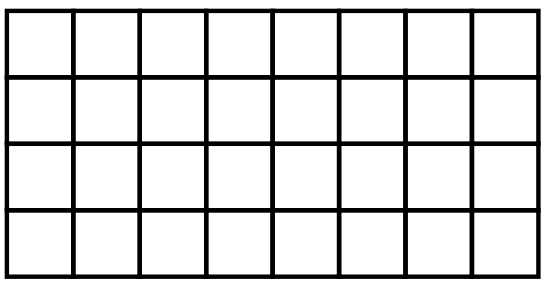
Variáveis locais

Definição

- As variáveis declaradas dentro de uma função são chamadas de variáveis locais e são conhecidas apenas dentro da função;
- Uma variável local existe apenas durante a execução do bloco de código onde ela foi declarada;
- A variável local é criada quando a função for chamada e destruída na saída da função;
- A variável local não é conhecida nas funções chamadas pela função onde ela foi definida;
- Duas variáveis com o mesmo nome mas declaradas em duas funções diferentes serão diferentes.

Passo a passo

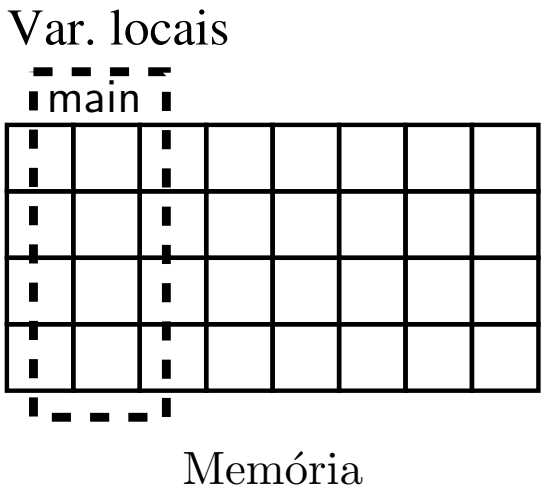
```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



Memória

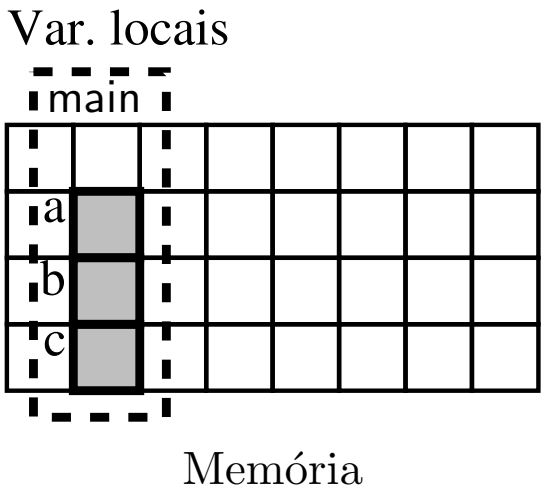
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



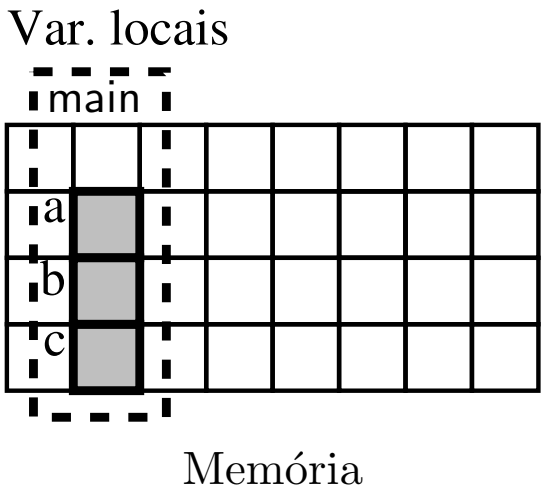
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



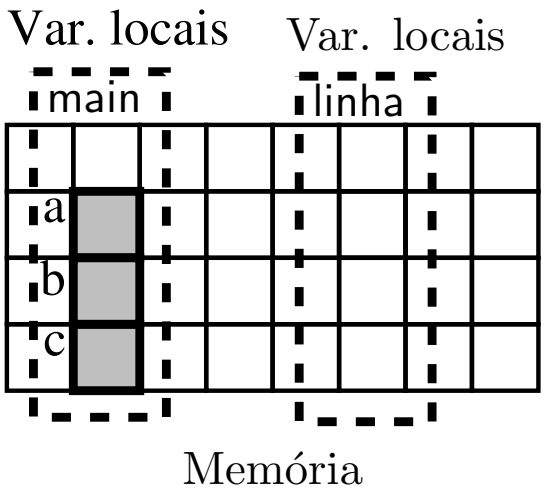
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



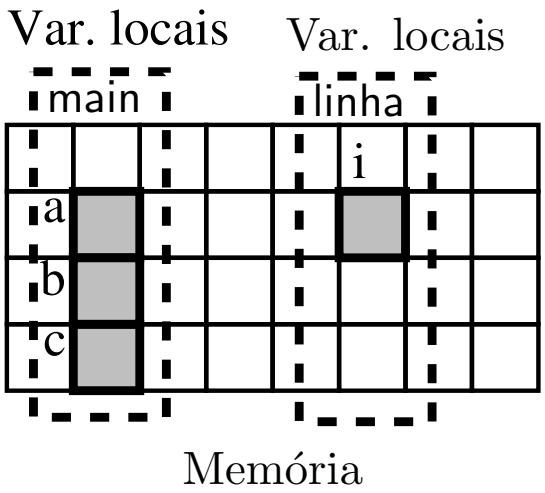
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



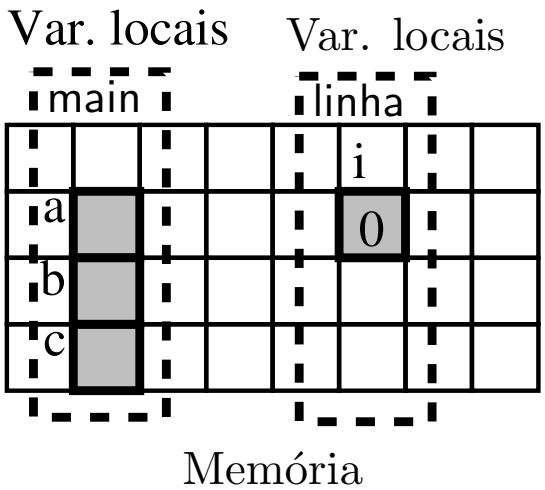
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



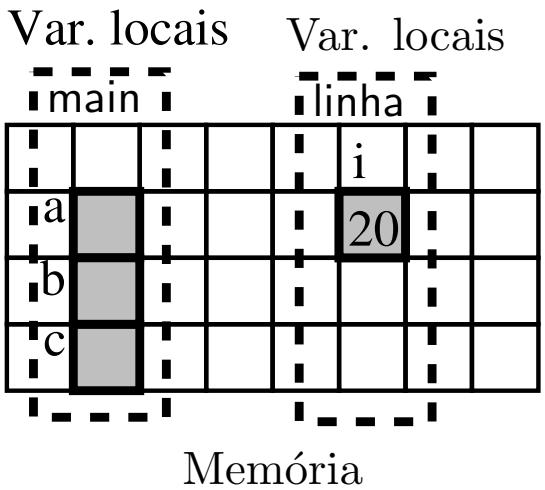
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



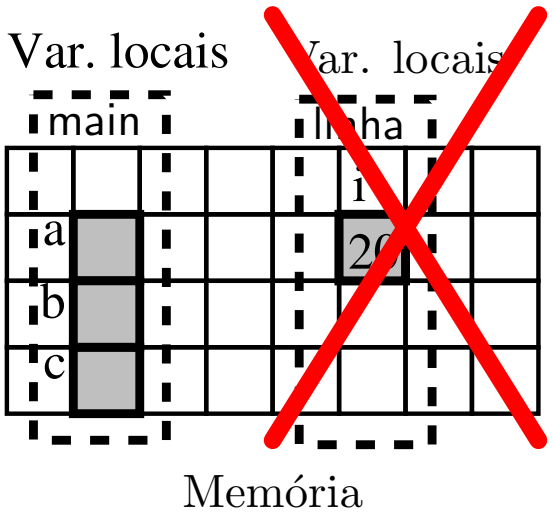
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



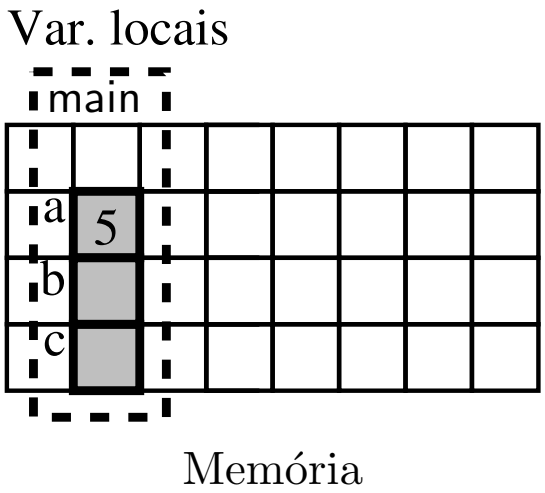
Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



Passo a passo

```
void linha(void)
{
  int i;
  for (i=0; i < 20; i++)
  {
    printf("_");
  }
  printf("\n");
}
int main(void)
{
  int a,b,c;
  linha(void);
  a = 5;
}
```



Exercício

É correto?

```
int inicializacao(void)
{
    a = 15;
    printf("a inicializada\n");
    return a;
}

int main(void)
{
    int a;
    a = inicializacao();
    printf("a=%d\n", a);
}
```

Exercício

É correto?

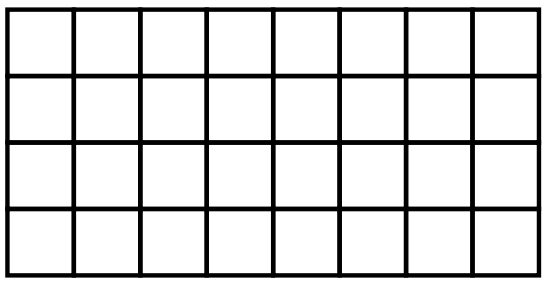
```
void inicializacao(void)
{
    int a;
    a = 15;
    printf("a inicializada\n");
    return a;
}

int main(void)
{
    a = inicializacao();
    printf("a=%d\n", a);
}
```

Exercício

```c
void inicializacao(void)
{
    int a;
    a = 15;
    return a;
}
int main(void)
{
    int a;
    a = inicializacao();
    printf("a inicializada\n");
    printf("a=%d\n", a);
}
```

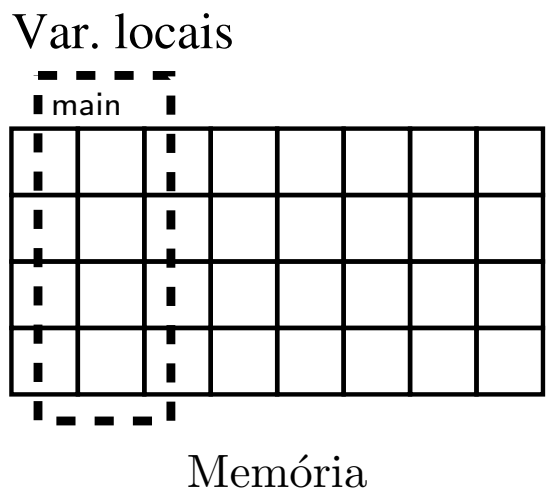
As variáveis `a` das funções `main` e `inicializacao` são diferentes. `a` do `main` recebe o valor de retorno da função.

```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```

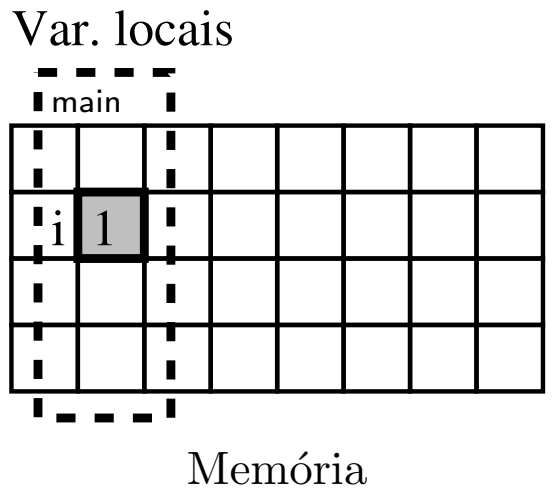


Memória

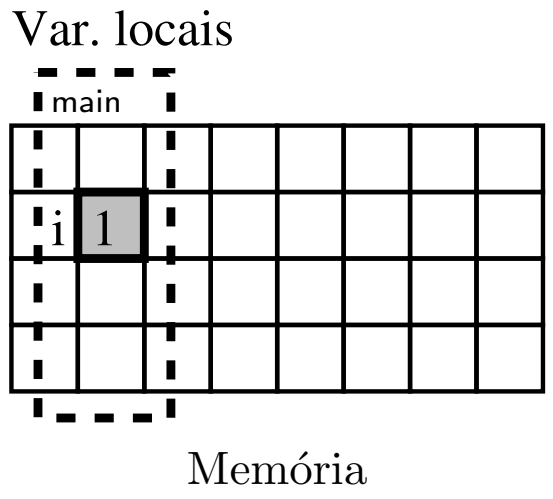
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



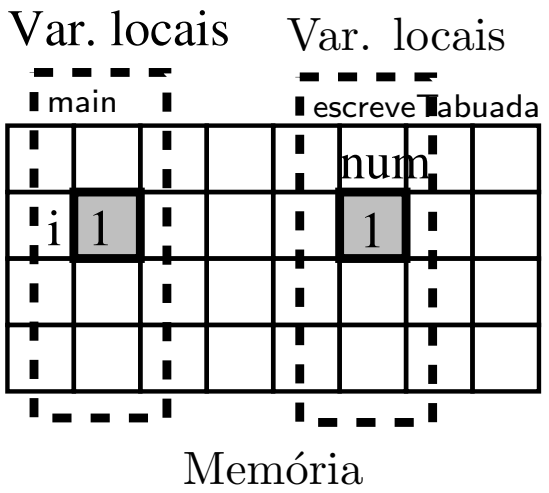
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



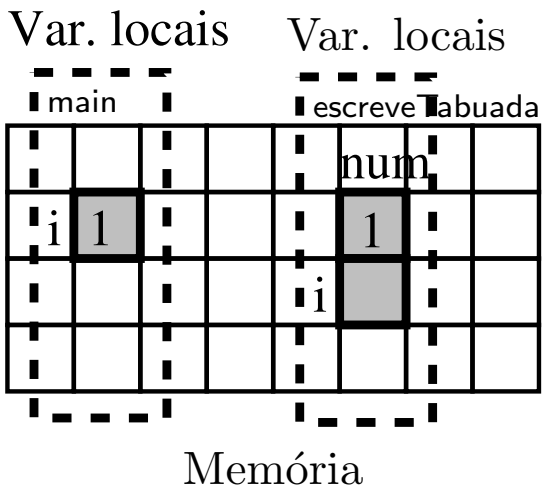
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



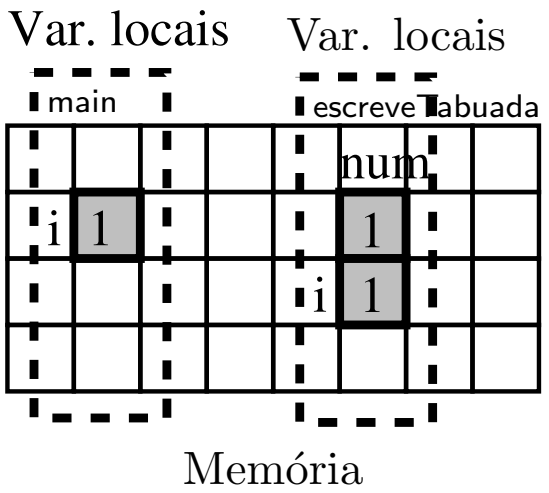
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



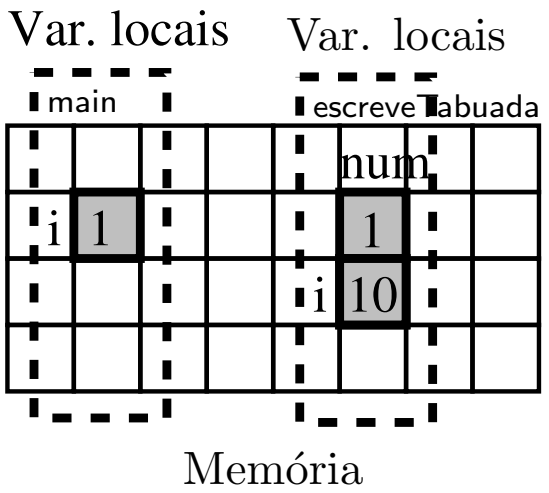
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



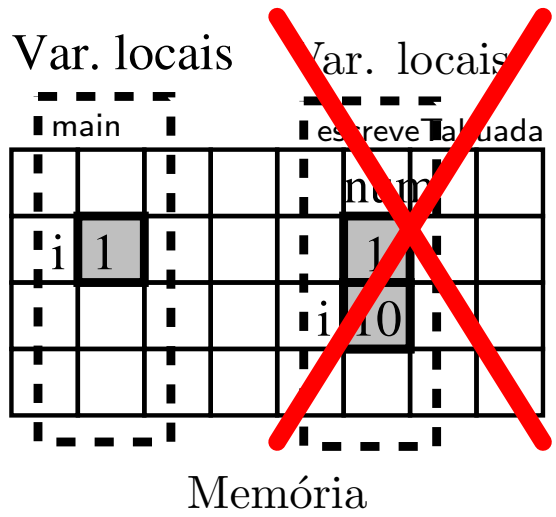
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



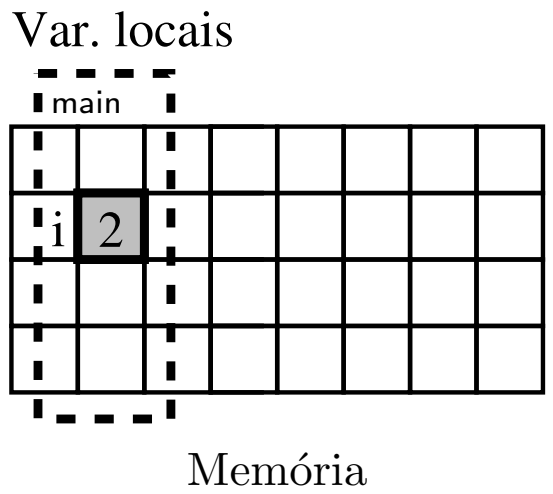
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



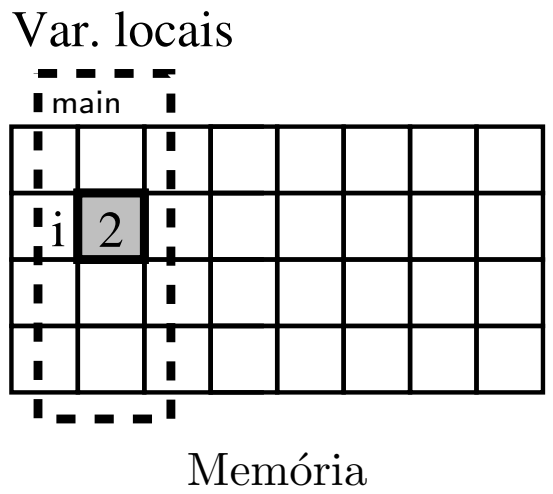
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



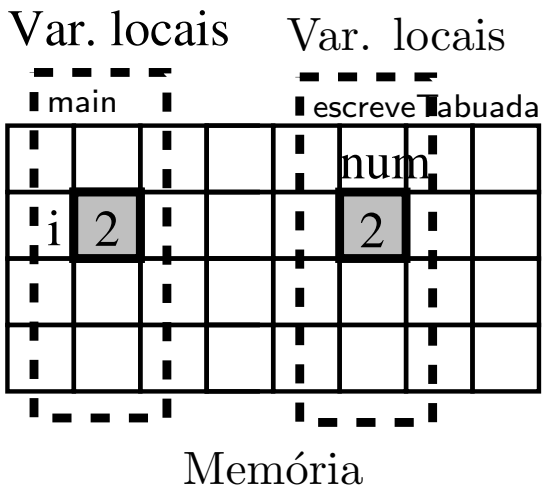
```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



```
void escreveTabuada(int num)
{
  int i;
  for (i=1; i<=10; i++)
    printf("%d x %d=%d\n",num,i,num*i);
  printf("\n");
}

int main(void)
{
  int i;
  for (i=1; i<=10; i++)
    escreveTabuada(i);
}
```



Variáveis globais

Definição

- É possível definir variáveis conhecidas de todo mundo;
- São chamadas variáveis globais;
- Declaração no cabeçalho do programa;
- Uso não aconselhado:
  - Ocupam memória;
  - Todo mundo pode modificar o valor!
  - Podem ser cobertas por outras variáveis

Orientação

Não use variáveis globais

Exemplo

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   inicializacao();
   printf("a=%d\n", a);
}
```

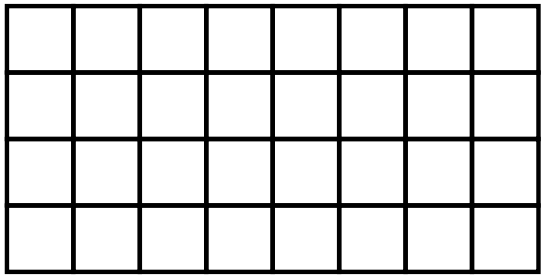
```
a inicializada
a=15
```

Cuidado! As variáveis `a` globais e locais (da função `main`) são diferentes.

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```

```
a inicializada
a=10
```

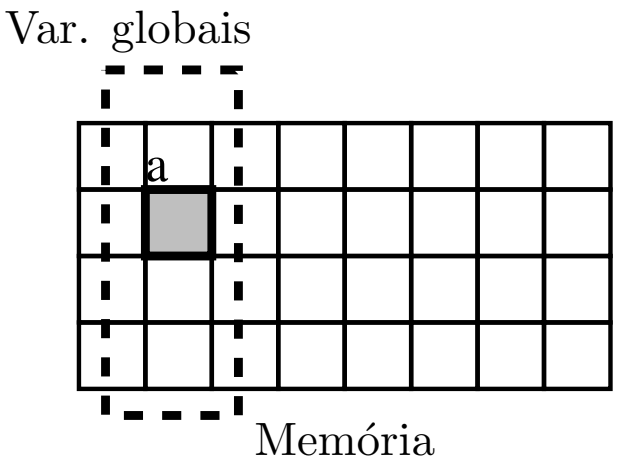
```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



Memória

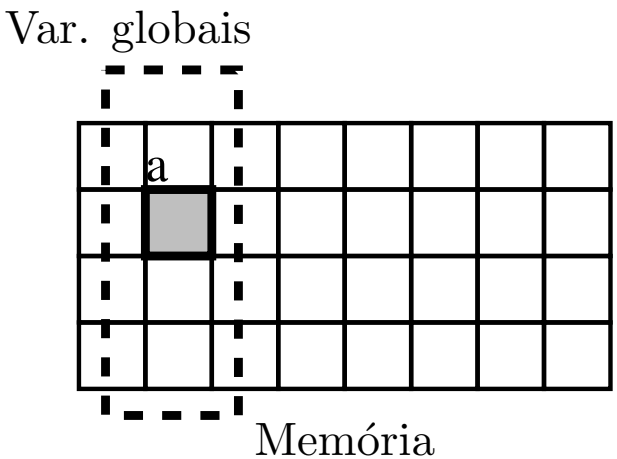
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



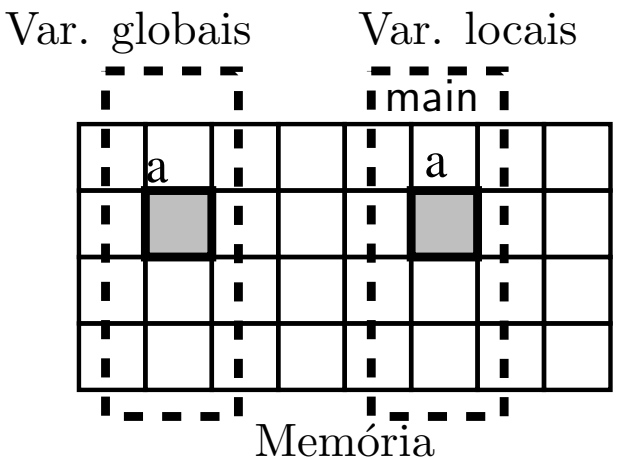
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



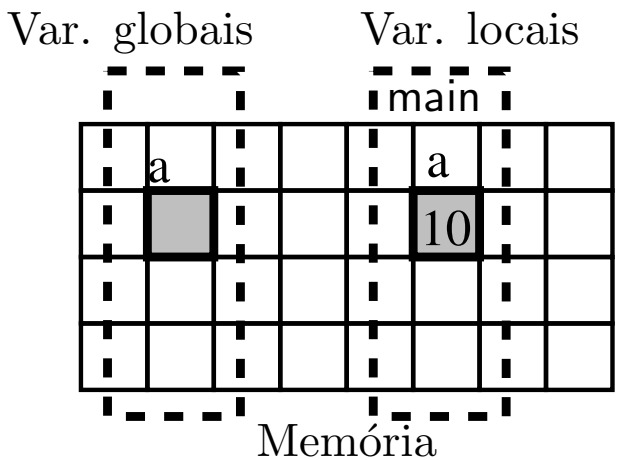
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



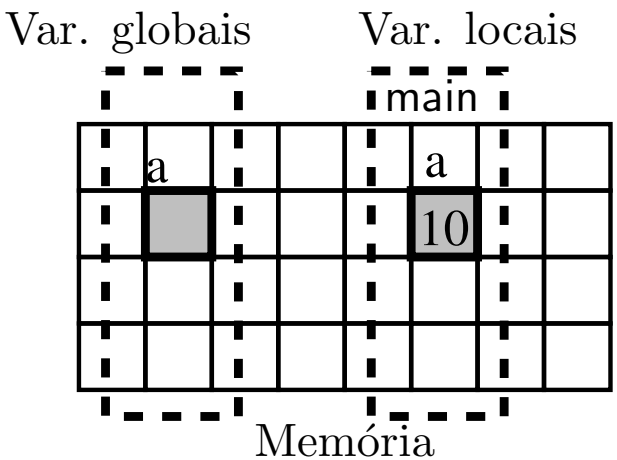
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



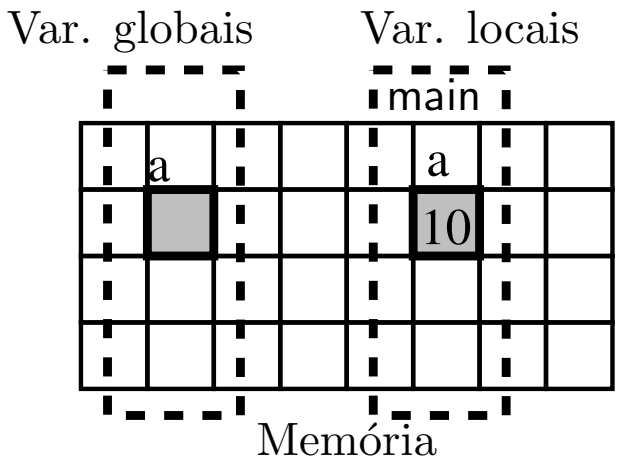
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



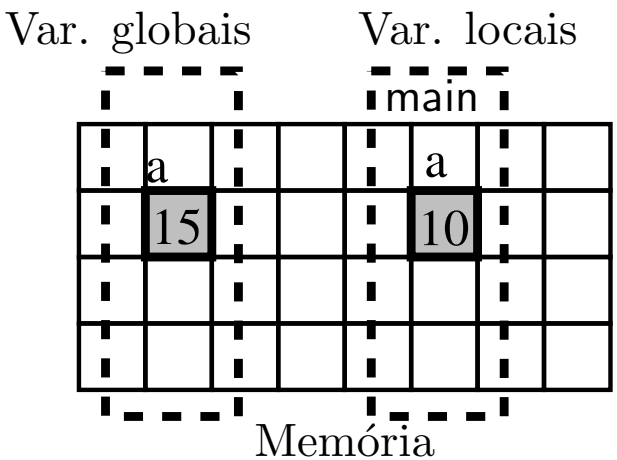
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



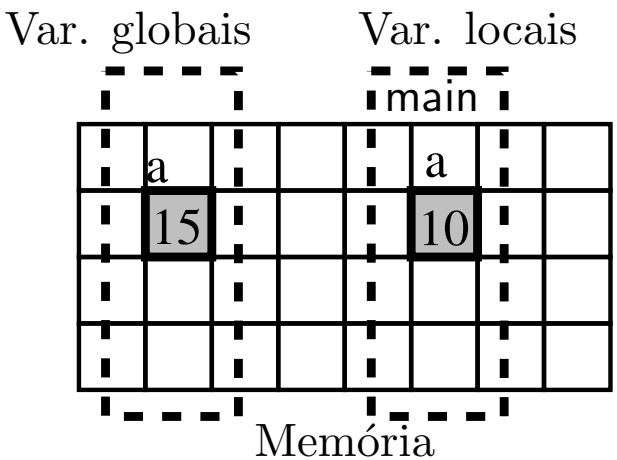
Tela

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



Tela

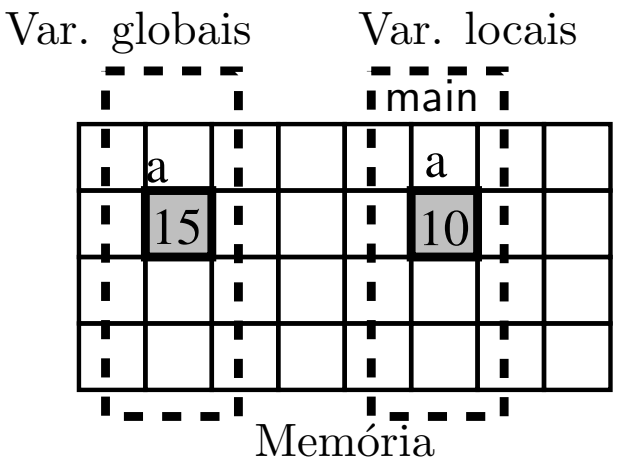
```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



Tela

```
a inicializada
```

```
int a;
void inicializacao(void)
{
   a = 15;
   printf("a inicializada\n");
}
int main(void)
{
   int a;
   a = 10;
   inicializacao();
   printf("a=%d\n", a);
}
```



Tela

```
a inicializada
a=10
```