



Ciência da Computação
Algoritmos e Estrutura de Dados 1

Memória e Funções

Rafael Liberato
liberato@utfpr.edu.br

Agenda

- O que é por que usar função
- Anatomia de uma função
- Detalhes de implementação
- Sugestão de passos para o desenvolvimento de uma função

O que é uma função

O que é e por que usar função

O que é

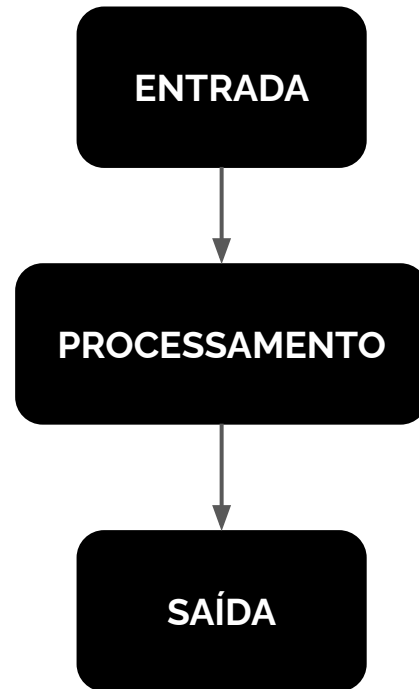
Funções são sub-rotinas

Benefícios

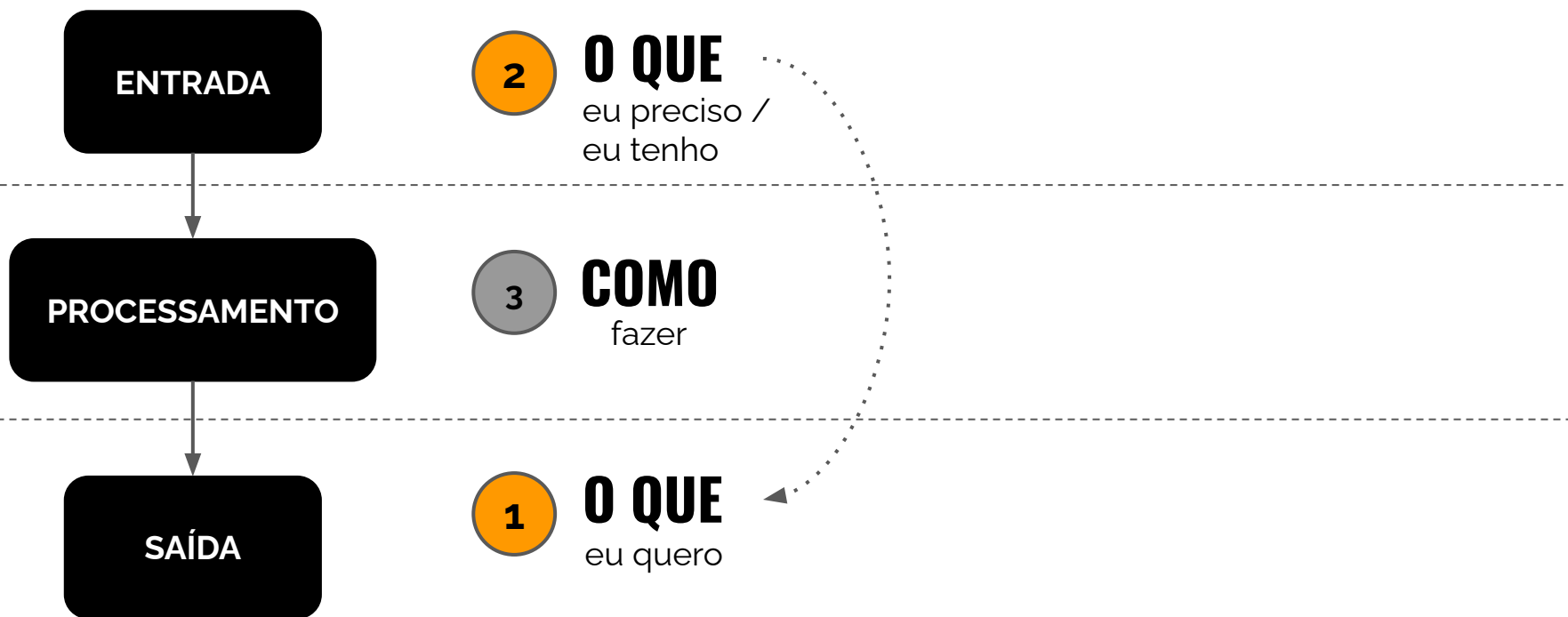
- Modularização/Organização.
- Facilita a compreensão
- Reutilização. Evita repetição de código
- Facilita a manutenção.
- Divisão de responsabilidade. Tarefas bem definidas

Anatomia de uma função

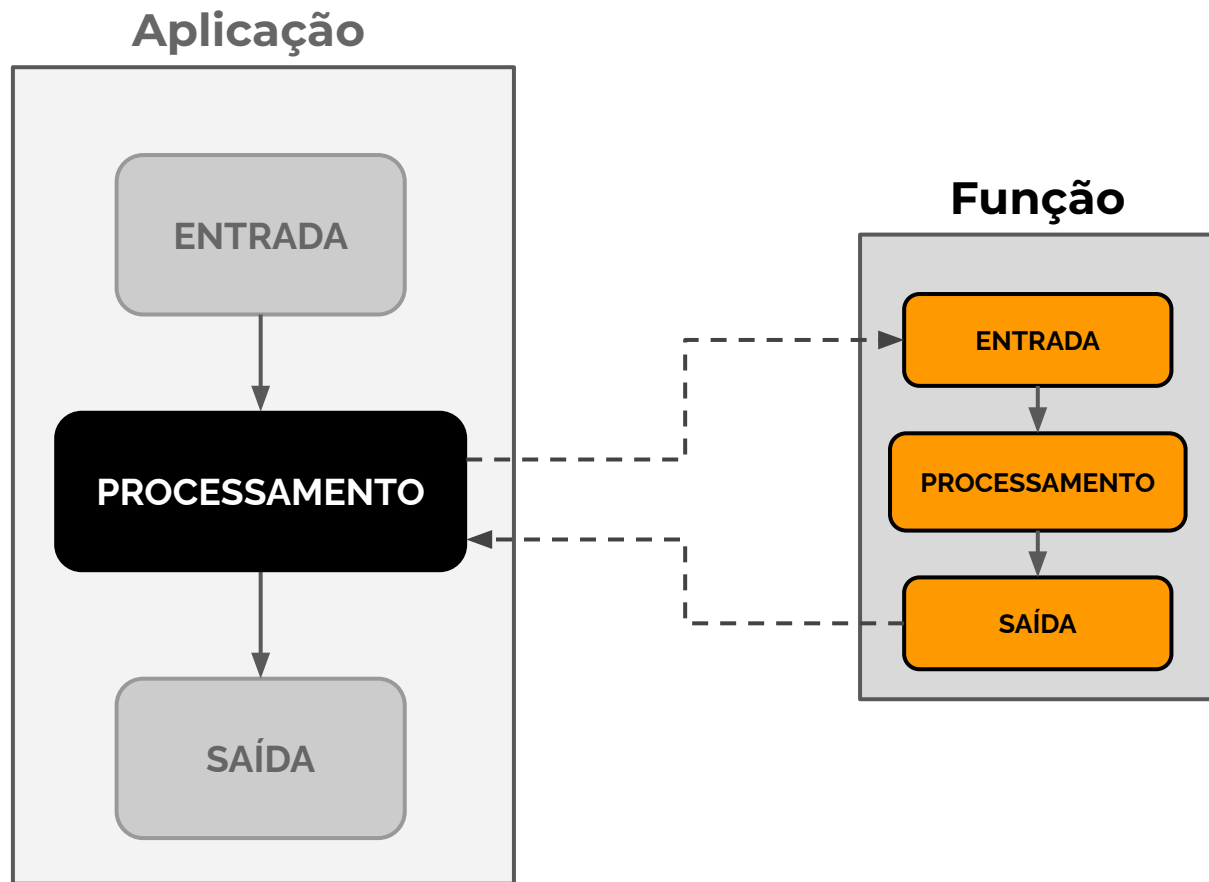
Anatomia de uma função



Anatomia de uma função



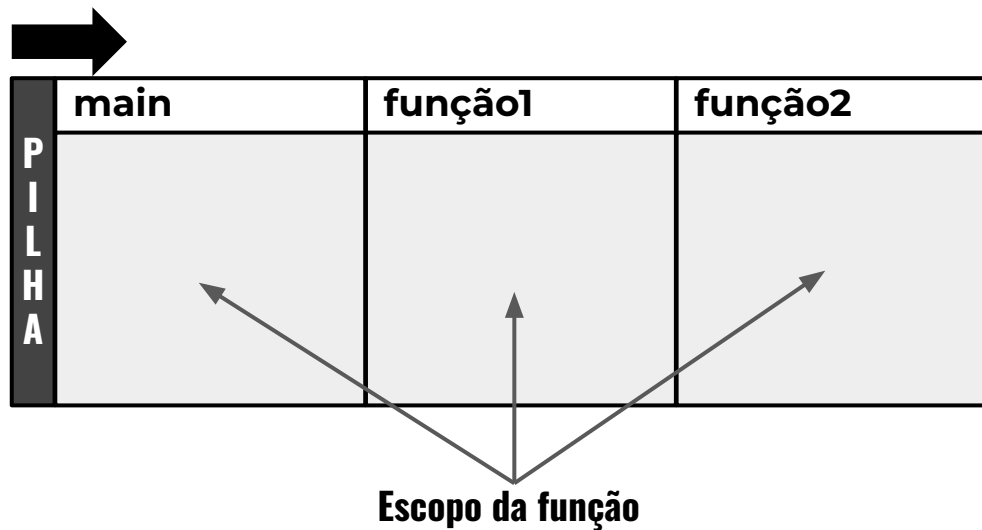
Anatomia de uma função



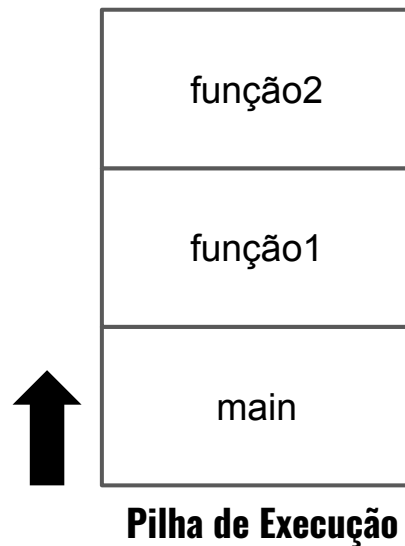
Anatomia de uma função

Comportamento

Antes de prosseguirmos com as diferentes formas de entrada e saída utilizadas nas funções, vamos relembrar o **comportamento** da execução das funções.



Uma função não tem acesso ao escopo da outra. Por exemplo, a **função1** não consegue “enxergar” os parâmetros e/ou variáveis locais declaradas em **função2** ou **main**.



Detalhes de Implementação

Detalhes de implementação

Diferentes formas de entrada e saída

ENTRADA

a) Parâmetros com valores **absolutos**

b) Parâmetros com valores **referência** (ponteiros)

PROCESSAMENTO

SAÍDA

a) Forma convencional usando **return**

b) Estratégia do scanf usando um **parâmetro com referência**

Nome atribuído por mim para facilitar a fixação do conceito

Entrada da Função

Passagem de parâmetro

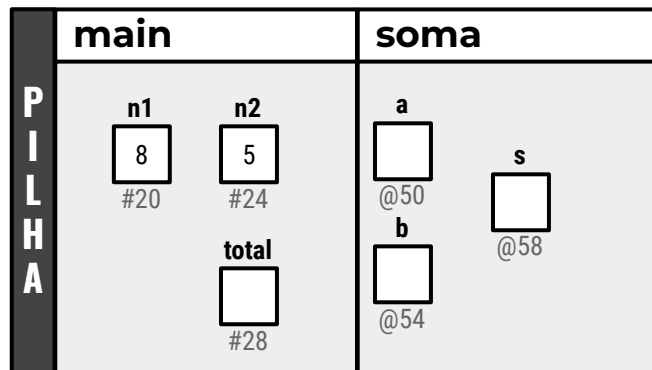
Detalhes de implementação

Passagem de Parâmetro

(Exemplo)
Função para somar 2 valores

Passando parâmetros
com **valor absoluto**

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(n1, n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```



Detalhes de implementação

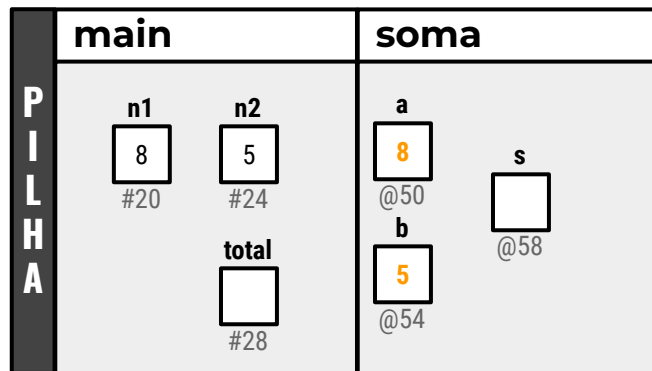
Passagem de Parâmetro

(Exemplo)
Função para somar 2 valores

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

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(n1, n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```

Passando parâmetros
com **valor absoluto**



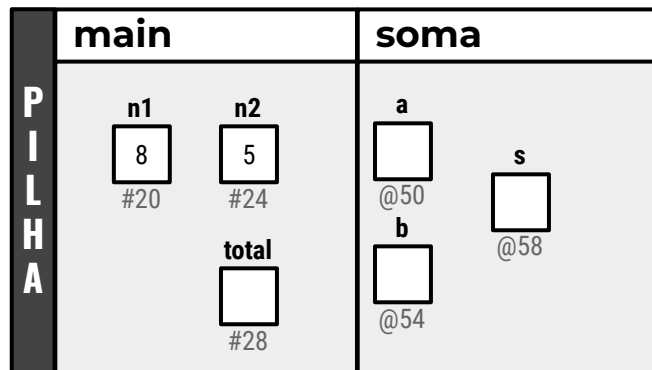
Detalhes de implementação

Passagem de Parâmetro

(Exemplo)
Função para somar 2 valores

Passando parâmetros
com **referência**

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(&n1, &n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```



Detalhes de implementação

Passagem de Parâmetro

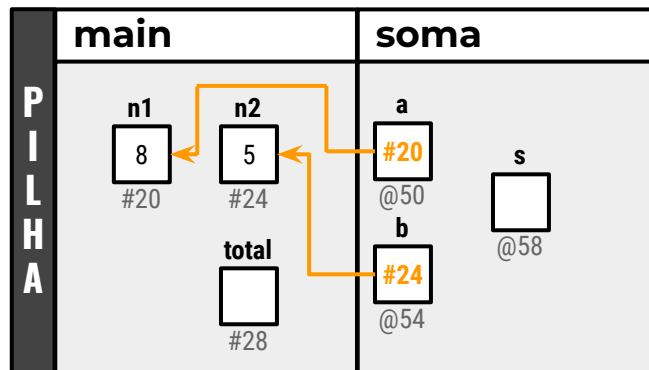
(Exemplo)

Função para somar 2 valores

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

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(&n1, &n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```

Passando parâmetros
com **referência**



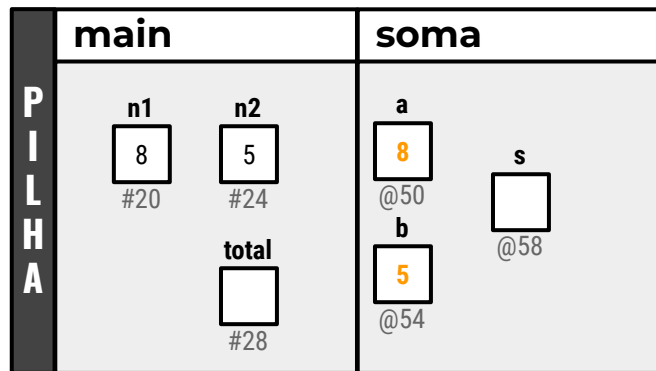
Detalhes de implementação

Passagem de Parâmetro

(Exemplo)

Função para somar 2 valores

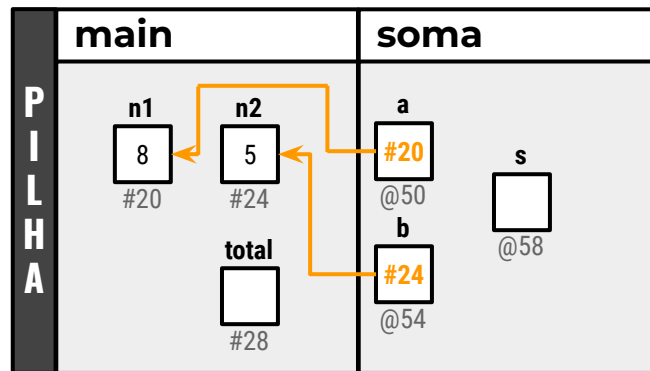
Passando parâmetros com
valor **absoluto**



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

```
int total = soma(n1, n2);
```

Passando parâmetros com
referência



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

```
int total = soma(&n1, &n2);
```

Saída da Função

Detalhes de implementação

Saída

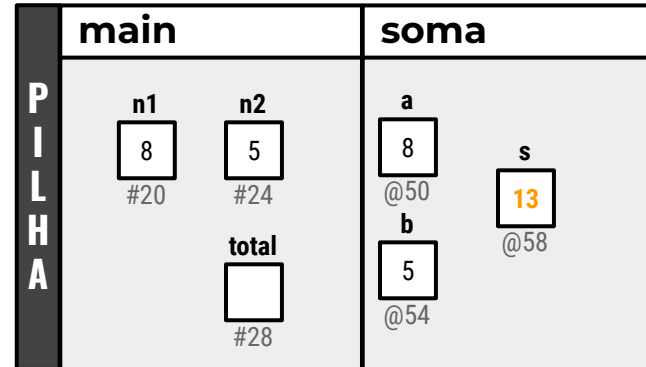
(Exemplo)

Função para somar 2 valores

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

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(n1, n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```

Forma convencional
usando **return**



Detalhes de implementação

Saída

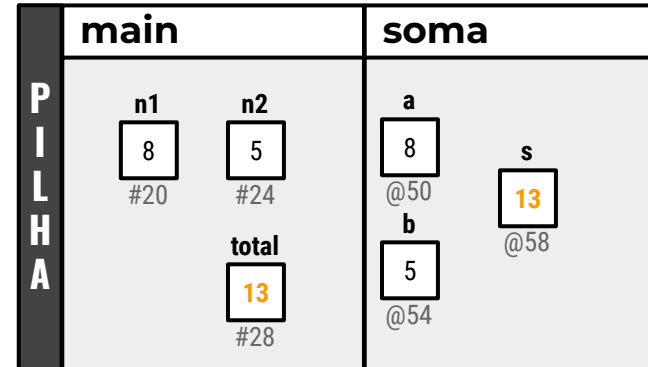
(Exemplo)

Função para somar 2 valores

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

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total = soma(n1, n2);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```

Forma convencional
usando **return**



*int total = soma(8, 5);
int total = 13;*

Detalhes de implementação

Saída

(Exemplo)

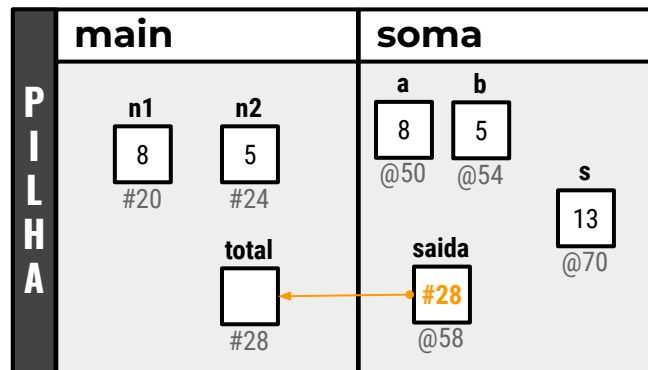
Função para somar 2 valores

Estratégia scanf usando parâmetros com referência

```
int main(){
    // Entrada
    int n1 = 8;
    int n2 = 5;

    // Processamento
    int total;
    soma(n1, n2, &total);

    // Saída
    printf("%d\n", total);
    return 0;
}
```



Detalhes de implementação

Saída

(Exemplo)

Função para somar 2 valores

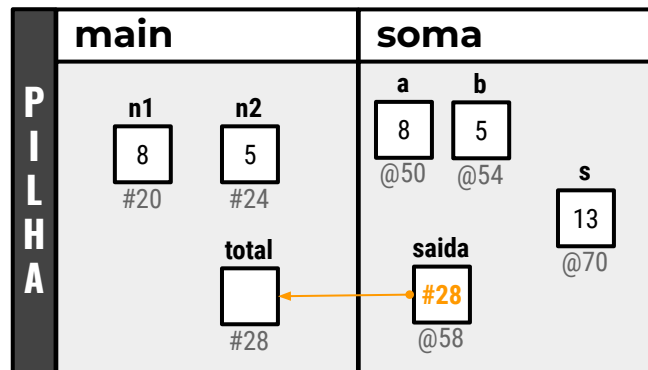
```
void soma(int a, int b, int* saida){  
    int s = a + b;
```

?

Estratégia scanf usando parâmetros com referência

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total;  
    soma(n1, n2, &total);
```

```
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```



Detalhes de implementação

Saída

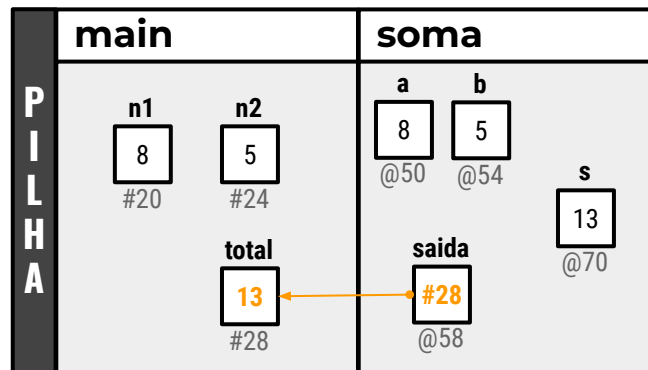
(Exemplo)

Função para somar 2 valores

```
void soma(int a, int b, int* saida){  
    int s = a + b;  
  
    *saida = s;  
}
```

Estratégia scanf usando parâmetros com referência

```
int main(){  
    // Entrada  
    int n1 = 8;  
    int n2 = 5;  
  
    // Processamento  
    int total;  
    soma(n1, n2, &total);  
  
    // Saída  
    printf("%d\n", total);  
    return 0;  
}
```



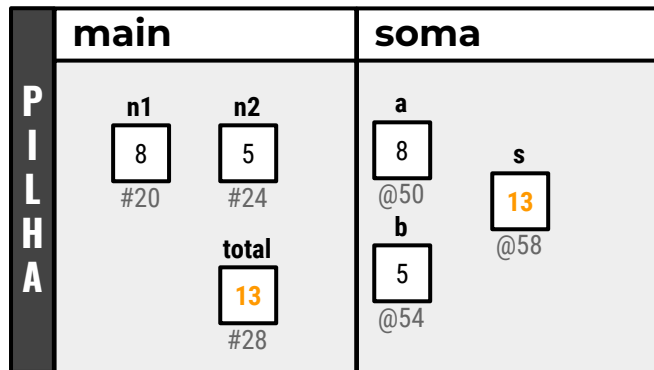
Detalhes de implementação

Saída

(Exemplo)

Função para somar 2 valores

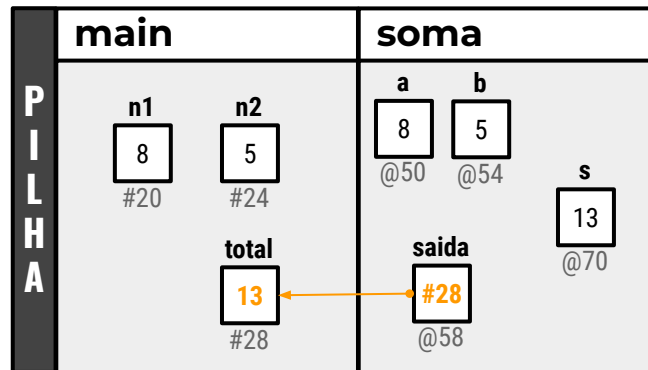
Forma convencional usando
return



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

```
total = soma(n1, n2);
```

Estratégia scanf usando
parâmetros com referência



```
void soma(int a, int b, int* saida){  
    int s = a + b;  
    *saida = s;  
}
```

```
soma(n1, n2, &total);
```

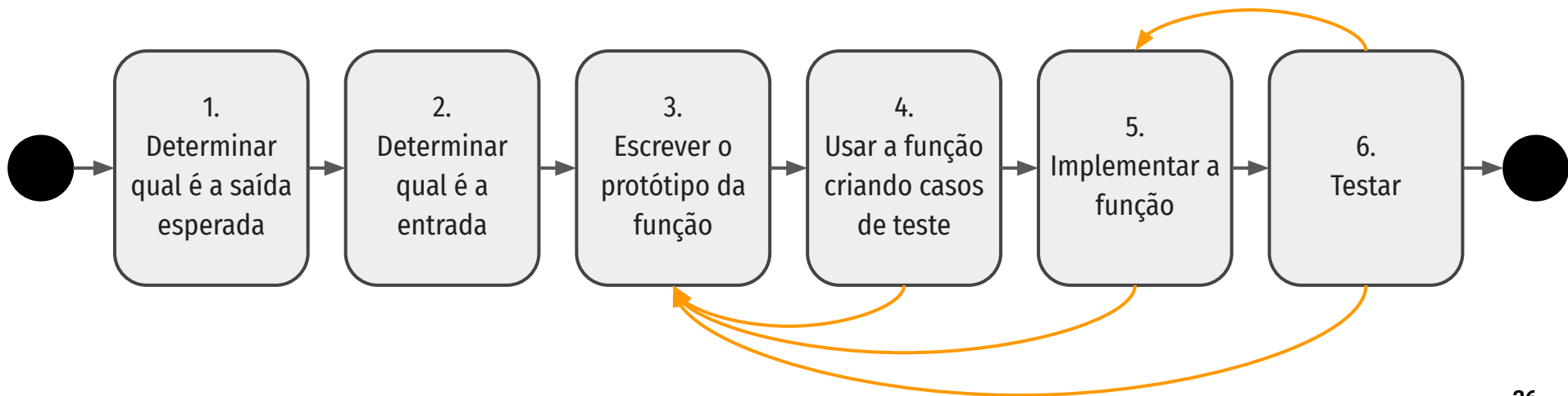

(Sugestão)

Etapas no desenvolvimento de uma função

Etapas no desenvolvimento da função

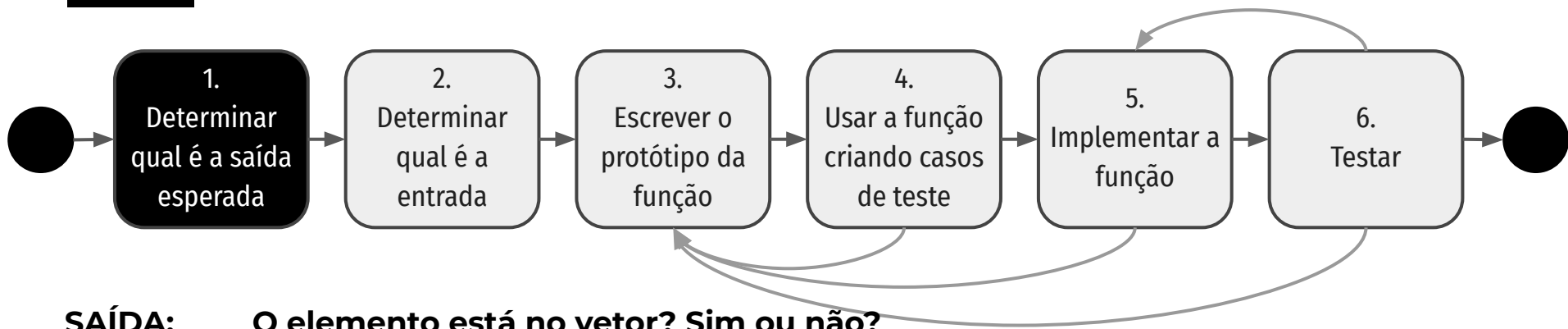
Sugestão

1. Determinar qual é a saída esperada
2. Identificar qual é a entrada necessária/disponível
3. Escrever o protótipo da função
4. Usar a função criando casos de teste
5. Implementar a função transformando a entrada na saída esperada
6. Testar



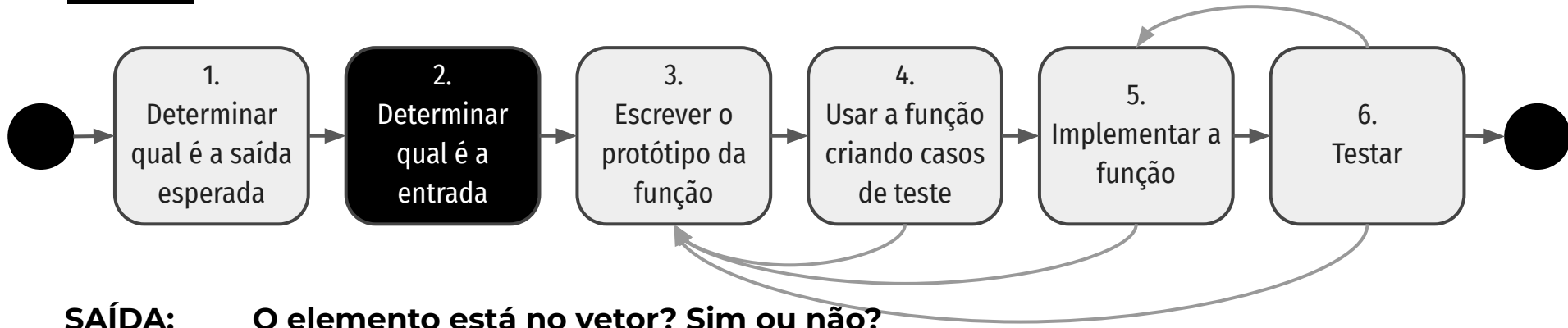
Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor

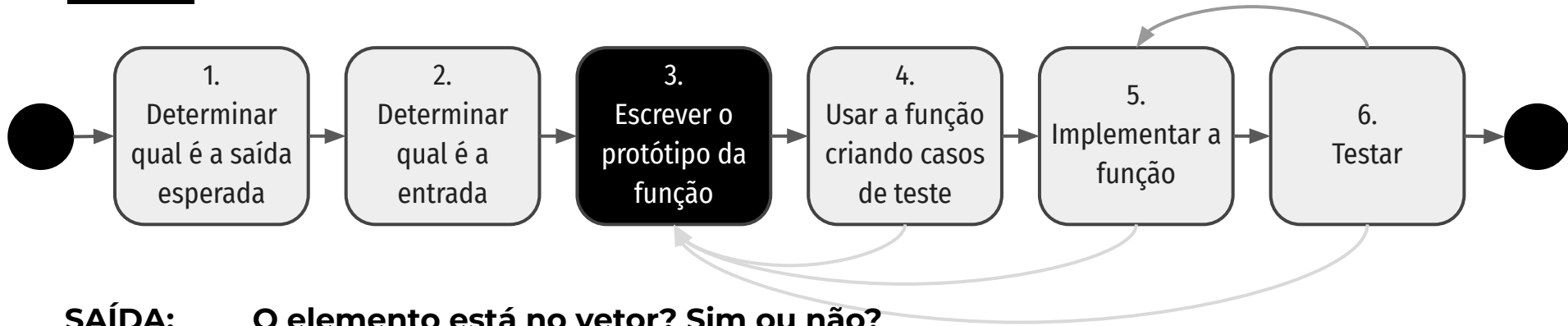


SAÍDA: O elemento está no vetor? Sim ou não?

ENTRADA: O vetor e o elemento a ser procurado

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor de inteiros



SAÍDA: O elemento está no vetor? Sim ou não?

ENTRADA: O vetor e o elemento a ser procurado

```
bool busca(int* v, int elemento);
```

```
int busca(int* v, int elemento);
```

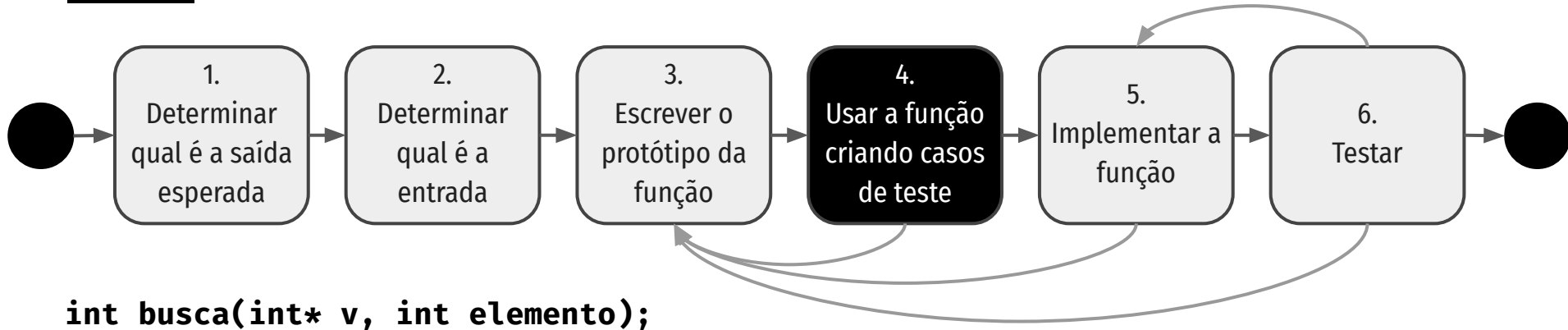
É necessário somente saber se o elemento está contido no vetor?

É necessário saber a posição do elemento encontrado?

O que será informado quando o elemento não for encontrado?

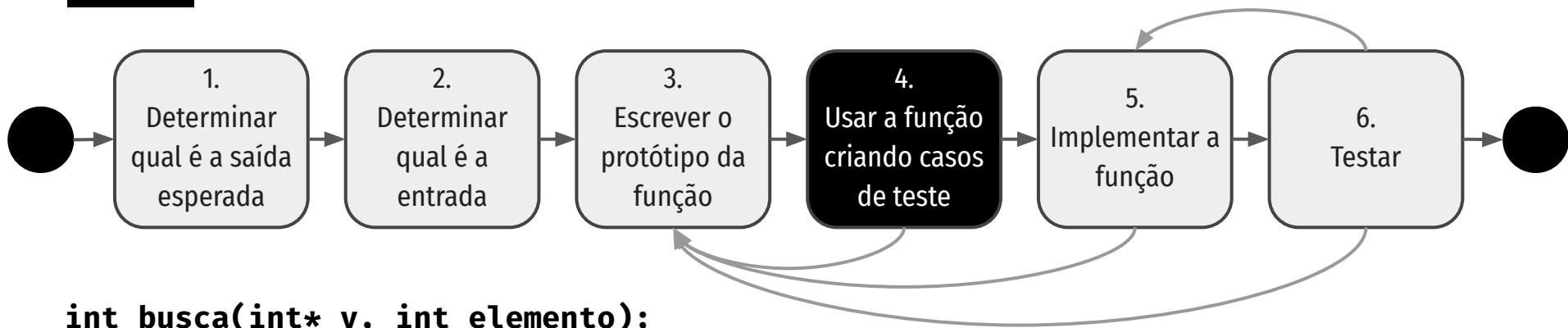
Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int elemento);
```

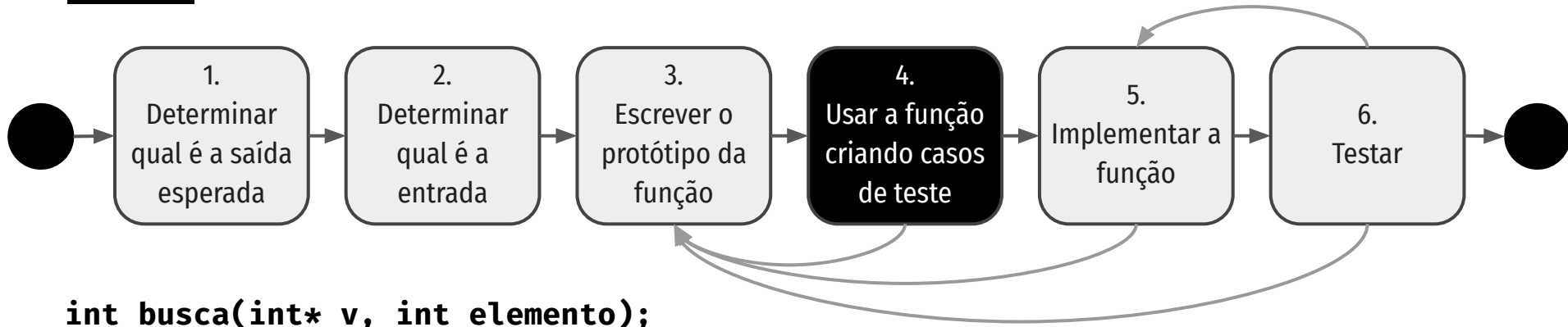
```
int main(){  
    int vet[5] = {10,20,30,40,50};
```

```
    printf("%d \n", busca(vet, 10)); // Saída Esperada: 0  
    printf("%d \n", busca(vet, 15)); // Saída Esperada: -1  
    printf("%d \n", busca(vet, 50)); // Saída Esperada: 4  
    printf("%d \n", busca(vet, 12)); // Saída Esperada: -1
```

```
}
```

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int elemento);
```

```
int main(){  
    int vet[5] = {10,20,30,40,50};
```

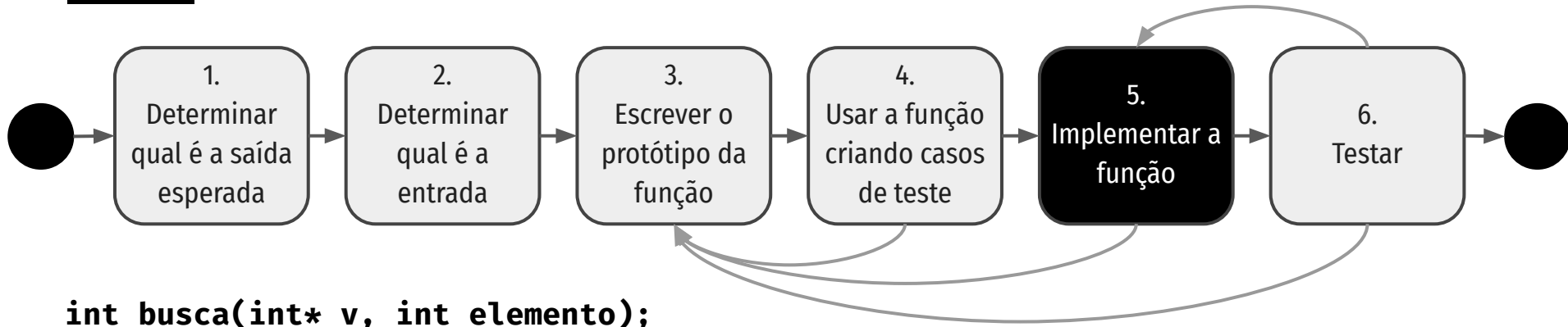
```
    printf("%d \n", busca(NULL, 10)); // Saída Esperada: -1
```

```
    printf("%d \n", busca(vet, -10)); // Saída Esperada: -1
```

```
}
```


Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor

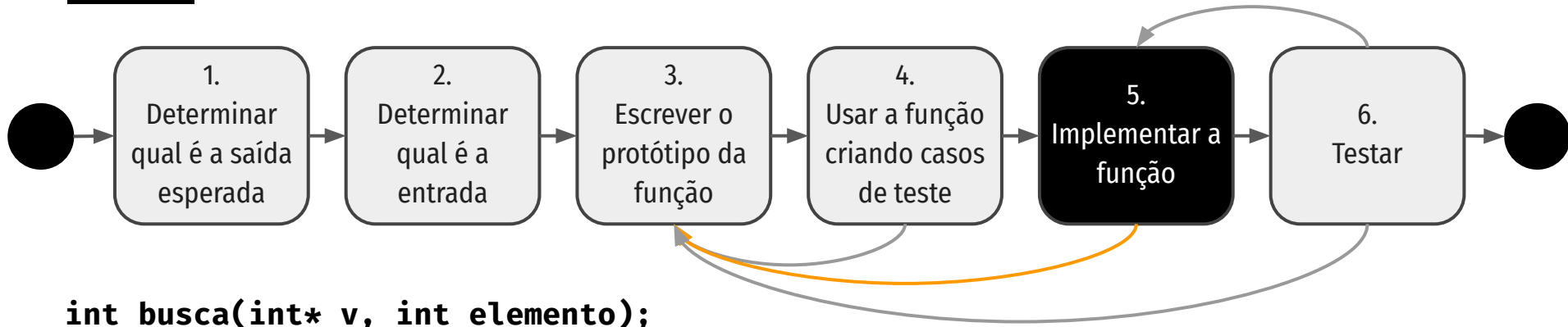


```
int busca(int* v, int elemento);
```

```
int busca(int* v, int elemento){  
    if (v == NULL) return -1;  
  
    int i;  
    for(i=0; i<  
}
```

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int elemento);
```

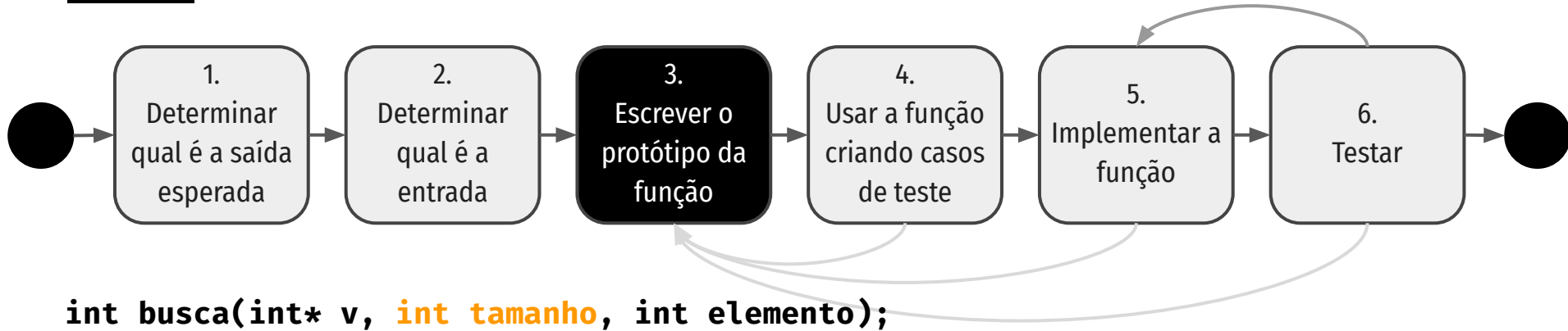
```
int busca(int* v, int elemento){  
    if (v == NULL) return -1;
```

```
    int i;  
    for(i=0; i< OPS... eu também preciso do tamanho do vetor  
}
```

Vamos voltar ao passo 3

Exemplo

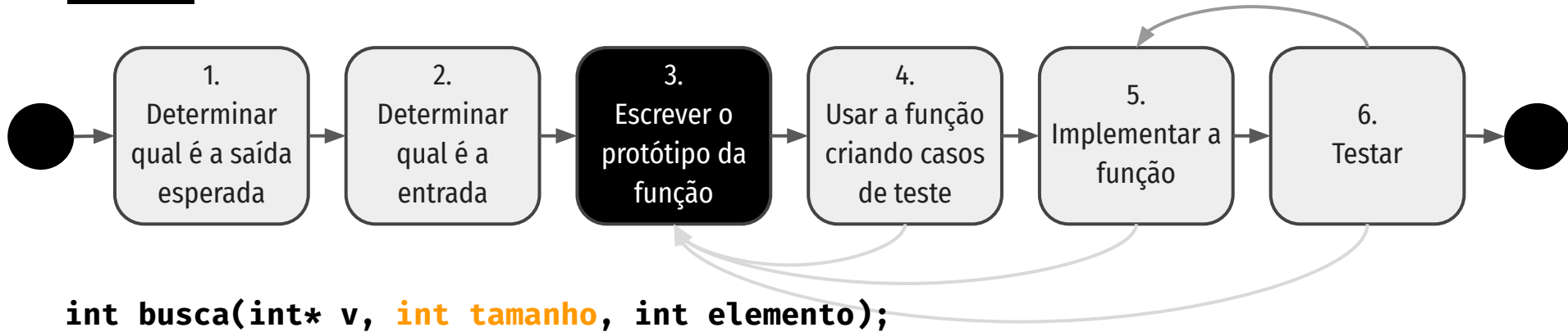
Escreva uma função para encontrar um determinado elemento em um vetor de inteiros



Agora vamos precisar atualizar os casos de teste no passo 4

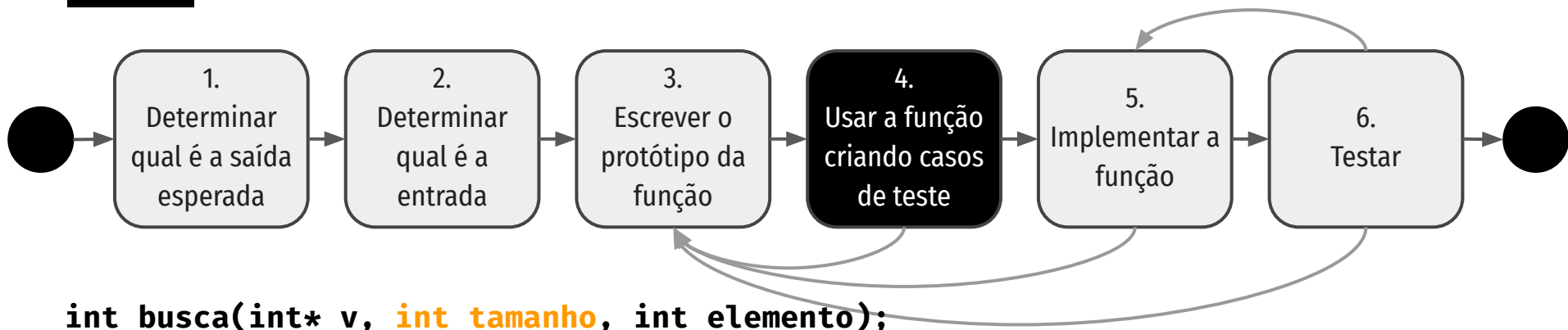
Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor de inteiros



Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int tamanho, int elemento);
```

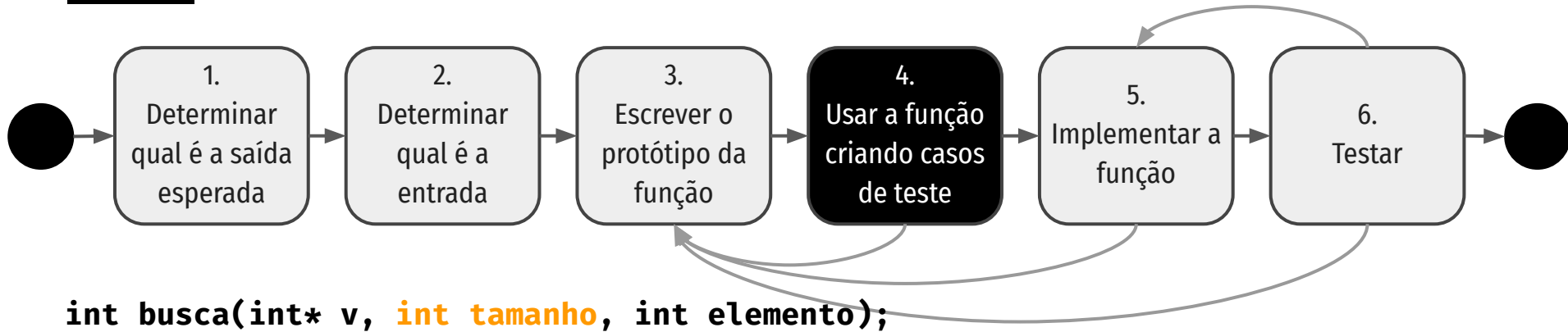
```
int main(){  
    int vet[5] = {10,20,30,40,50};
```

```
    printf("%d \n", busca(vet, 5, 10)); // Saída Esperada: 0  
    printf("%d \n", busca(vet, 5, 15)); // Saída Esperada: 1  
    printf("%d \n", busca(vet, 5, 50)); // Saída Esperada: 4  
    printf("%d \n", busca(vet, 5, 12)); // Saída Esperada: -1
```

```
}
```

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int tamanho, int elemento);
```

```
int main(){  
    int vet[5] = {10,20,30,40,50};
```

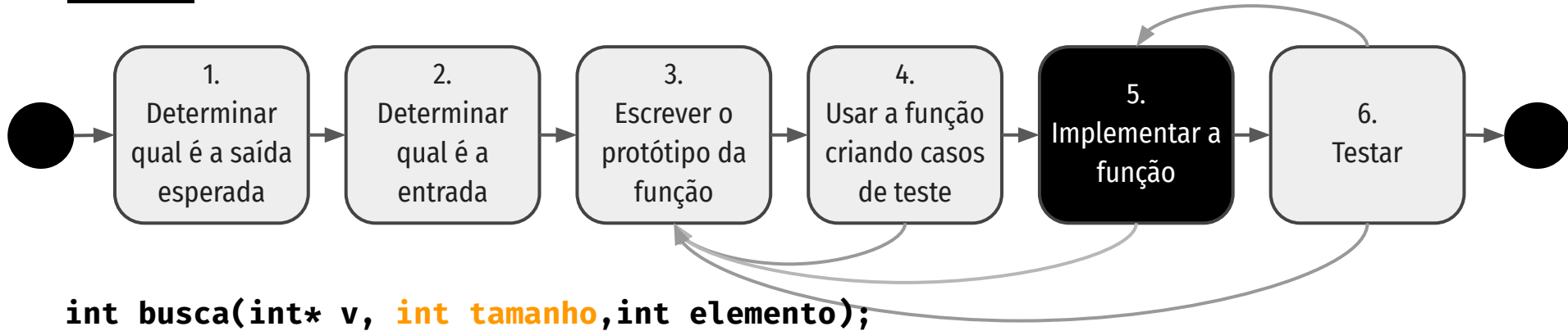
```
    printf("%d \n", busca(NULL, 5, 10)); // Saída Esperada: -1
```

```
    printf("%d \n", busca(vet, 5, -10)); // Saída Esperada: -1
```

```
}
```

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



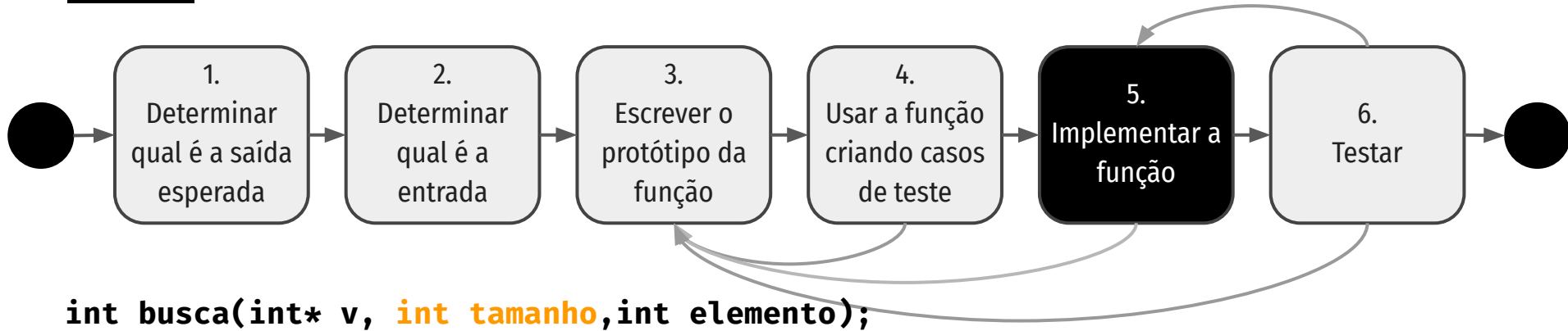
```
int busca(int* v, int tamanho, int elemento);
```

```
int busca(int* v, int elemento){  
    if (v == NULL) return -1;
```

```
    int i;  
    for(i=0; i< OPS... eu também preciso do tamanho do vetor  
}
```

Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor

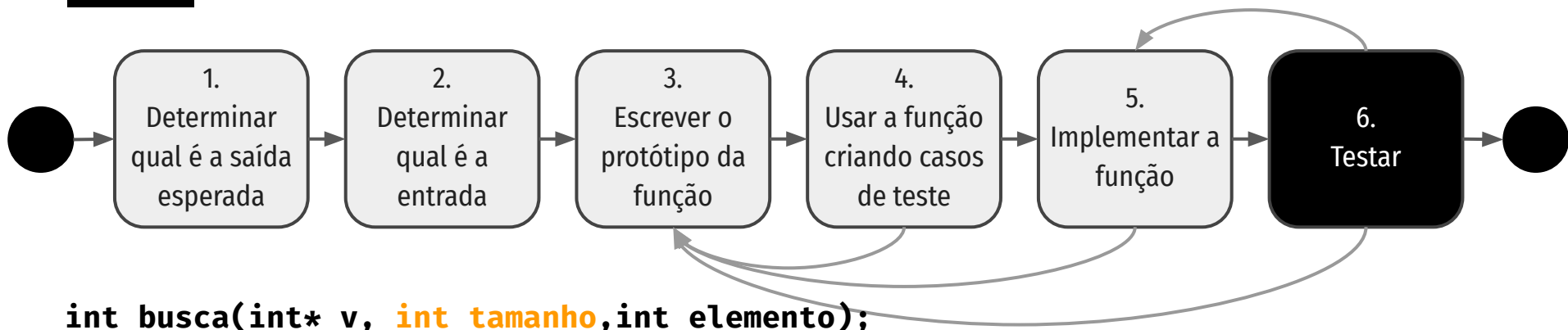


```
int busca(int* v, int tamanho, int elemento);
```

```
int busca(int* v, int tamanho, int elemento){  
    if (v == NULL) return -1;  
  
    int i;  
    for(i=0; i<tamanho; i++){  
        if(v[i] == elemento) return i;  
    }  
    return -1;  
}
```


Exemplo

Escreva uma função para encontrar um determinado elemento em um vetor



```
int busca(int* v, int tamanho, int elemento);
```

```
int busca(int* v, int elemento){
    if (v == NULL) return -1;

    int i;
    for(i=0; i<tamanho; i++){
        if(v[i] == elemento) return i;
    }
    return -1;
}
```

Nesse passo precisamos somente executar os casos de teste escritos no **passo 4**.

Caso alguma saída não tenha o comportamento esperado, voltamos para implementação no **passo 5** ou para a escrita do protótipo no **passo 3**.

Jim