
Raciocínio Algorítmico e Fundamentos da Computação

— Danilo Rios —
09/05/2025

Função

- Quando começamos a programar falamos da função main
- Também falamos que função é um bloco de código que realiza determinada funcionalidade
- Utilizamos as funções: printf e scanf

Função

- Agora vamos falar sobre as nossas funções

Função

- Como criar uma função?

Função

- Como criar uma função?
- `<tipo de retorno> <nome da função> (<parâmetros recebidos>) {
 <código da função>
}`

Tipo de retorno

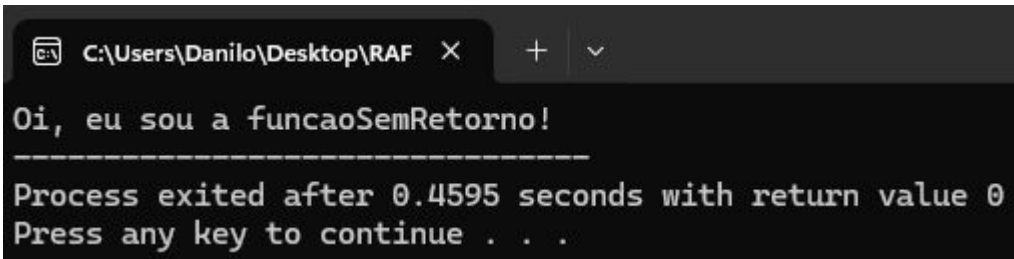
- void
 - Não tem retorno
- Tipo de variável
 - int, float, double, char
 - Retorna o tipo informado

Função sem retorno

```
1  #include <stdio.h>
2
3  void funcaoSemRetorno() {
4      printf("Oi, eu sou a funcaoSemRetorno!");
5  }
6
7  int main() {
8
9      funcaoSemRetorno();
10
11     return 0;
12 }
```

Função sem retorno

```
1  #include <stdio.h>
2
3  void funcaoSemRetorno() {
4      printf("Oi, eu sou a funcaoSemRetorno!");
5  }
6
7  int main() {
8
9      funcaoSemRetorno();
10
11     return 0;
12 }
```



C:\Users\Danilo\Desktop\RAF X + v

Oi, eu sou a funcaoSemRetorno!

Process exited after 0.4595 seconds with return value 0

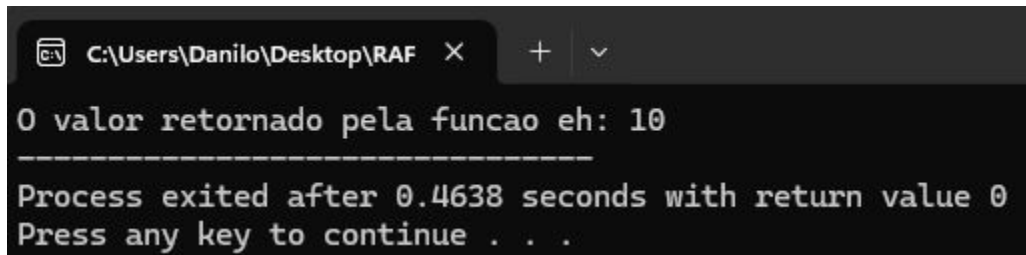
Press any key to continue . . .

Função com retorno

```
1  #include <stdio.h>
2
3  int funcaoRetornaValor10() {
4      return 10;
5  }
6
7  int main() {
8
9      int valorRetornado = funcaoRetornaValor10();
10     printf("O valor retornado pela funcao eh: %d", valorRetornado);
11
12     return 0;
13 }
```

Função com retorno

```
1  #include <stdio.h>
2
3  int funcaoRetornaValor10() {
4      return 10;
5  }
6
7  int main() {
8
9      int valorRetornado = funcaoRetornaValor10();
10     printf("O valor retornado pela funcao eh: %d", valorRetornado);
11
12     return 0;
13 }
```



C:\Users\Danilo\Desktop\RAF X + v

O valor retornado pela funcao eh: 10

Process exited after 0.4638 seconds with return value 0

Press any key to continue . . .

Parâmetros recebidos

- O que é parâmetro?

Parâmetros recebidos

- O que é parâmetro?
 - É uma informação que a função precisa receber

Parâmetros recebidos

```
1  #include <stdio.h>
2
3  int funcaoSoma(int numero1, int numero2) {
4      return numero1 + numero2;
5  }
6
7  int main() {
8
9      int valorRetornado = funcaoSoma(2,2);
10     printf("O valor retornado pela funcao eh: %d", valorRetornado);
11
12     return 0;
13 }
```

ou

```
int funcaoSoma(int numero1, int numero2) {
    int somatoria = numero1 + numero2;
    return somatoria;
}
```

Parâmetros recebidos

```
1  #include <stdio.h>
2
3  int funcaoSoma(int numero1, int numero2) {
4      return numero1 + numero2;
5  }
6
7  int main() {
8
9      int valorRetornado = funcaoSoma(2,2);
10     printf("O valor retornado pela funcao eh: %d", valorRetornado);
11
12     return 0;
13 }
```

ou

```
int funcaoSoma(int numero1, int numero2) {
    int somatoria = numero1 + numero2;
    return somatoria;
}
```

```
C:\Users\Danilo\Desktop\RAF x + v
0 valor retornado pela funcao eh: 4
-----
Process exited after 0.4734 seconds with return value 0
Press any key to continue . . .
```

Outro exemplo

- O que vai acontecer ao executar?

```
1  #include <stdio.h>
2
3  int main() {
4      int valorRetornado = funcaoSubtracao(10,5);
5      printf("O valor retornado pela funcao eh: %d", valorRetornado);
6
7      return 0;
8  }
9
10
11 int funcaoSubtracao(int numero1, int numero2) {
12     return numero1 - numero2;
13 }
```

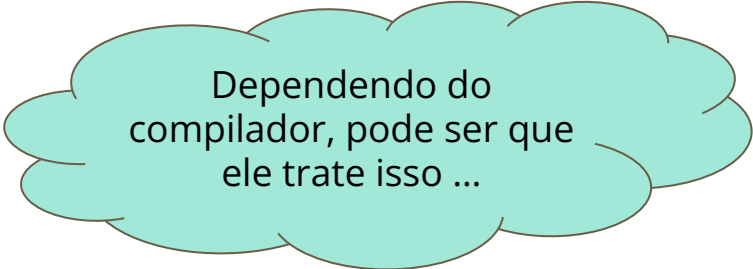
Outro exemplo

```
1  #include <stdio.h>
2
3  int main() {
4      int valorRetornado = funcaoSubtracao(10,5);
5      printf("O valor retornado pela funcao eh: %d", valorRetornado);
6
7      return 0;
8  }
9
10
11 int funcaoSubtracao(int numero1, int numero2) {
12     return numero1 - numero2;
13 }
```

Para minha
surpresa
funcionou...

```
C:\Users\Danilo\Desktop\RAF x + v
O valor retornado pela funcao eh: 5
-----
Process exited after 0.4695 seconds with return value 0
Press any key to continue . . .
```


Ordem das funções



Dependendo do compilador, pode ser que ele trate isso ...

- O código só consegue utilizar uma função se ela já existe
 - Se a função está implementada antes da linha de código que está utilizando
 - Então todas as funções devem ser implementadas antes da função main
 - Funções que são utilizadas por outras funções também precisam ser implementadas antes da sua utilização

Ordem das funções

- E se eu tiver a função A que utiliza a função B e dependendo do fluxo que está sendo executado da função B ela utiliza a função A?
 - Como que implementa uma antes da outra?
- Ou no caso de uma função que utiliza ela mesma?
 - Como que implementa uma função, por inteiro, antes dela mesmo se utilizar?
- Existe uma solução
 - Para esses casos problemáticos e também para não ter que se preocupar

Ordem das funções

- Podemos inserir o protótipo da função no começo do código e fazer a implementação em qualquer ordem do código
- Para declarar o protótipo de uma função:
 - <tipo de retorno> <nome função> (<parâmetros recebidos>);

Ordem das funções

```
1  #include <stdio.h>
2
3  //protótipo de função, implementação depois
4  int funcaoSubtracao(int numero1, int numero2);
5
6  int main() {
7      int valorRetornado = funcaoSubtracao(10,5);
8      printf("O valor retornado pela funcao eh: %d", valorRetornado);
9
10     return 0;
11 }
12
13
14 int funcaoSubtracao(int numero1, int numero2) {
15     return numero1 - numero2;
16 }
```

Função main

- Aprendemos e utilizamos a versão mais simples da função main

```
1  #include <stdio.h>
2
3  int main() {
4      //código fonte
5      return 0;
6  }
```

Função main

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4
5      //argc = quantidade de argumentos
6      //argv = valores recebidos
7
8      //verifica se recebeu 3 argumentos
9      if (argc != 3) {
10         printf("Erro! Nao recebi os 2 argumentos esperados.");
11         //nome do arquivo (automático) + 2 números
12         return 1;
13     }
14
15     //converte para int
16     int n1 = atoi(argv[1]);
17     int n2 = atoi(argv[2]);
18
19     printf("Numeros recebidos por parametro: %d e %d", n1, n2);
20
21     return 0;
22 }
```

- Ela também possui parâmetros

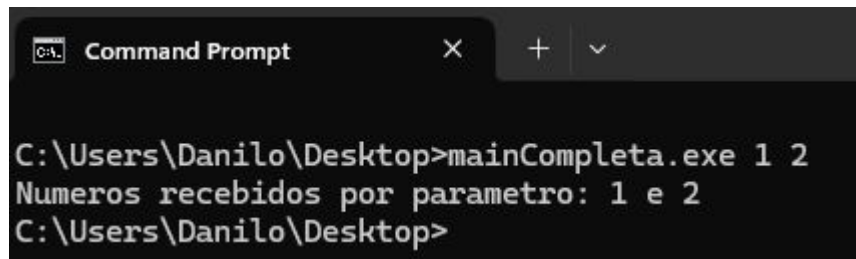
Função main

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4
5      //argc = quantidade de argumentos
6      //argv = valores recebidos
7
8      //verifica se recebeu 3 argumentos
9      if (argc != 3) {
10         printf("Erro! Nao recebi os 2 argumentos esperados.");
11         //nome do arquivo (automático) + 2 números
12         return 1;
13     }
14
```

```
C:\Users\Danilo\Desktop\RAF x + v
Erro! Nao recebi os 2 argumentos esperados.
-----
Process exited after 0.1613 seconds with return value 1
Press any key to continue . . .
o: %d e %d", n1, n2);
21     return 0;
22 }
```

Função main

- Como passamos argumentos, valores, para os parâmetros da main?
 - Quando executamos através do terminal/console/prompt de comando
 - Arquivo executável e os argumentos separados por espaço



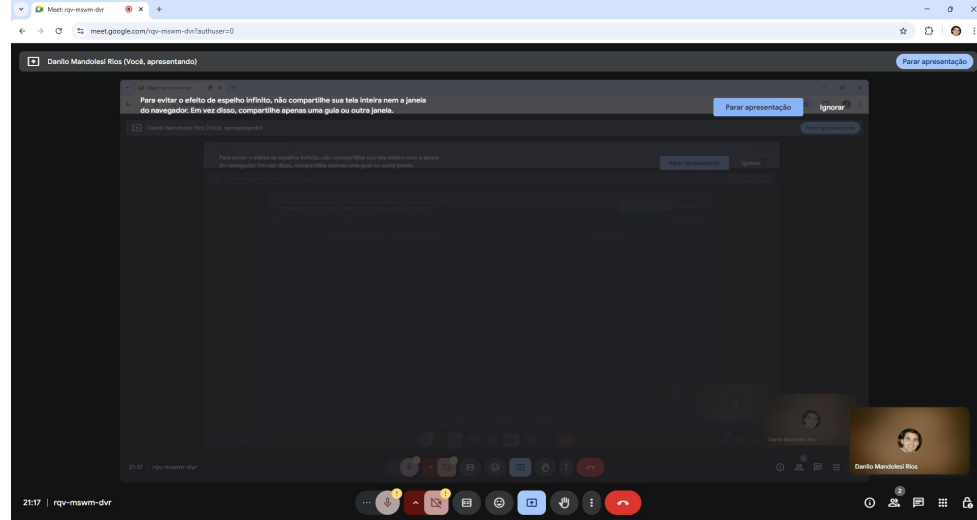
```
C:\Users\Danilo\Desktop>mainCompleta.exe 1 2
Numeros recebidos por parametro: 1 e 2
C:\Users\Danilo\Desktop>
```


Recursão

- O que é?

Recursão

- O que é?
 - É quando algo chama a si mesmo
 - Ex.: No Google Meet compartilhar a tela de compartilhar
 - Começa mostrando 1 tela
 - Depois está mostrando a tela compartilhada com 1 tela compartilhada
 - Depois está mostrando a tela compartilhada com 1 tela compartilhada mostrando também 1 tela compartilhada
 -



Recursão

- Em programação também temos recursão
- É quando uma função no meio do seu código chama ela mesma

Recursão

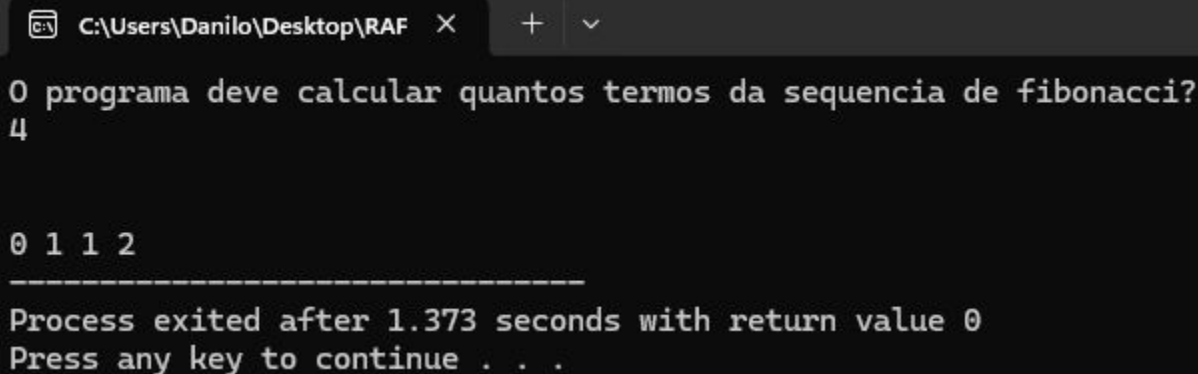
- O clássico de recursão é calcular o fatorial
 - Vamos deixar esse para o exercício no moodle
- Vamos ver com a Sequência de Fibonacci
 - 0, 1 (fixos)
 - Os próximos números são a somatória dos 2 números anteriores
 - 0, 1, $1_{(0+1)}$, $2_{(1+1)}$, $3_{(1+2)}$, $5_{(2+3)}$, $8_{(3+5)}$, $13_{(5+8)}$, $21_{(8+13)}$, ...

Re

```
1 #include <stdio.h>
2
3 int fibonacci(int termo) {
4     //os resultados fixos
5     if (termo == 0 || termo == 1) {
6         return termo;
7     }
8
9     //recursão para calcular o fibonacci
10    return fibonacci(termo - 2) + fibonacci(termo - 1);
11 }
12
13 int main() {
14     int qtd;
15
16     printf("O programa deve calcular quantos termos da sequencia de fibonacci?\n");
17     scanf("%d", &qtd);
18     printf("\n\n");
19
20     if (qtd > 0) {
21         int i;
22         for (i = 0; i < qtd; i++) {
23             printf("%d ", fibonacci(i));
24         }
25     } else {
26         printf("Nao eh possivel calcular 0 termos ou uma quantidade negativa de termos");
27     }
28
29     return 0;
30 }
```

Re

```
1 #include <stdio.h>
2
3 int fibonacci(int termo) {
4     //os resultados fixos
5     if (termo == 0 || termo == 1) {
6         return termo;
7     }
8
9     //recursão para calcular o fibonacci
10    return fibonacci(termo - 2) + fibonacci(termo - 1);
11 }
12
13 int main() {
14     int qtd;
15
16     printf("O programa deve calcular quantos termos da sequencia de fibonacci?\n");
17     scanf("%d", &qtd);
18     printf("\n\n");
19
20     if (qtd < 0) {
21         printf("O número de termos deve ser maior ou igual a 0.\n");
22     }
23
24     if (qtd > 0) {
25         printf("Sequencia de fibonacci com %d termos:\n", qtd);
26         printf("0 1 1 2\n");
27         printf("-----\n");
28     }
29
30     return 0;
}
```



```
C:\Users\Danilo\Desktop\RAF >
O programa deve calcular quantos termos da sequencia de fibonacci?
4
0 1 1 2
-----
Process exited after 1.373 seconds with return value 0
Press any key to continue . . .
```

e termos");

Recursão

- fibonacci(0);
 - return 0;
- fibonacci(1);
 - return 1;
- fibonacci(2);
 - fibonacci(0) + fibonacci(1);
 - fibonacci(0);
 - return 0;
 - fibonacci(1);
 - return 1;
 - return 1;

```
1 #include <stdio.h>
2
3 int fibonacci(int termo) {
4     //os resultados fixos
5     if(termo == 0 || termo == 1) {
6         return termo;
7     }
8
9     //recursão para calcular o fibonacci
10    return fibonacci(termo - 2) + fibonacci(termo - 1);
11 }
```

- fibonacci(3);
 - fibonacci(1) + fibonacci(2);
 - fibonacci(1);
 - return 1;
 - fibonacci(2);
 - fibonacci(0) + fibonacci(1);
 - fibonacci(0);
 - return 0;
 - fibonacci(1);
 - return 1;
 - return 1;
 - return 2;

Recursão

- Atenção com o “loop” infinito
 - Parecido com o while e do while que precisamos criar no código algum comando que muda o valor da verificação do loop
 - No caso da recursão também é necessário alterar o valor para não dar problema
 - ```
int fibonacci(int termo) {
 return fibonacci(termo); //nunca vai terminar de calcular
}
```



# Exercício 24

- Criar um código
  - Na função main
    - Pede um número
    - Se o número for 1, chama a função que imprime o texto
      - Sou a função do número 1
    - Se o número for 2, chama a função que imprime o texto
      - Sou a função do número 2
    - Se o número não for 1 ou 2, repete o pedido para digitar um número
- Enviar o exercício pelo Moodle
  - Conteúdo: arquivo .c

# Exercício 25

- Criar um código
  - Na função main
    - Recebe 1 número e chama a função que calcula e retorna o fatorial do número
    - Imprimir
      - O fatorial de <número> é <valor calculado>
  - Tem uma função que recebe 1 número por parâmetro, calcula o fatorial e retorna o resultado
- Enviar o exercício pelo Moodle
  - Conteúdo: arquivo .c

# Perguntas?

# Obrigado!

Até a próxima aula!