



Programação em Linguagem C

Funções

Estrutura básica função main()

main.c

```
1 #include <stdio.h>
2
3 int main()●
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
10
```

Programa principal (main)

Todo programa em linguagem C é estruturado em uma ou mais

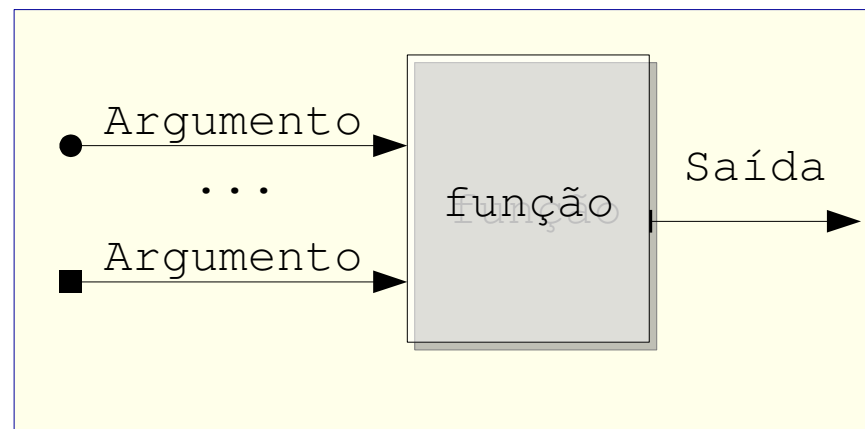
Funções.

O nome das funções é de escolha do programador, exceto a única função obrigatória, **main**.

Estrutura básica de uma função

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
10
```



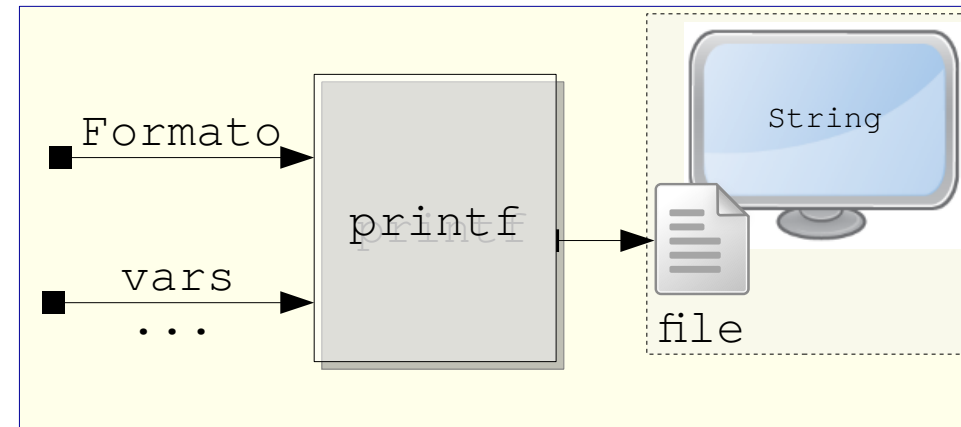
Estrutura de uma função

```
<tipo_saída> nome_da_funcao(<tipo_entrada> argumento,...)
{
    início do corpo da função
    corpo da função com retorno de dado
}
fim do corpo da função
```

Função printf()

source code

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("String");
6     return 0;
7 }
8
```



printf

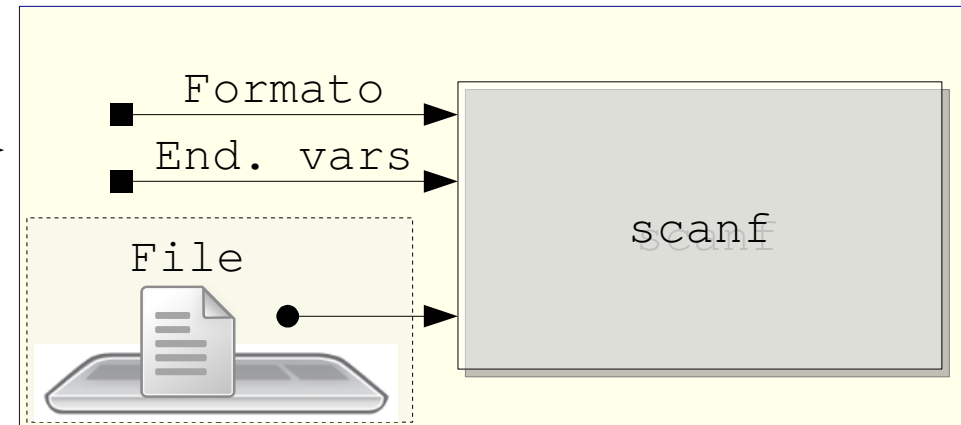
Imprime uma cadeia de caracteres no arquivo de saída padrão.
A saída padrão, geralmente, é o monitor.

Memória

...	'S'	't'	'r'	'i'	'n'	'g'	\0	...
-----	-----	-----	-----	-----	-----	-----	----	-----

Função scanf()

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      int num = 0;
6      printf("Numero: %d\n", num );
7      scanf("%d", &num );
8      printf("Numero: %d\n", num );
9      return 0;
10 }
11
```



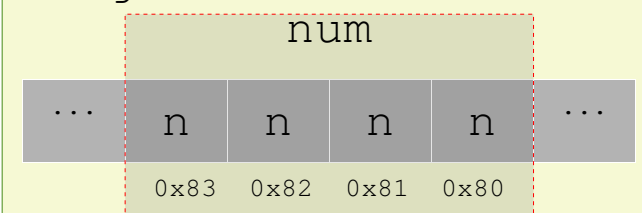
scanf()

Lê a entrada padrão, que geralmente, é o teclado

%d: tipo de dado de entrada: inteiro

&<var>: endereço da variável que receberá o dado lido.

Segmento da memória



$$74 = 0b01001010 = 0x4A$$

Função soma()

main.c

```
1 // Declaração de bibliotecas
2 #include <stdio.h>
3 // Declaração da função soma()
4 int soma( int a, int b )
5 {
6     return(a+b);
7 }
8 // Programa principal
9 int main()
10 {
11     // Declaração de variáveis
12     int parcela1;
13     int parcela2;
14     int total;
15
16     // Entrada de dados
17     printf("Digite dois numeros: ");
18     scanf("%d %d", &parcela1, &parcela2 );
19
20     // Execução
21     total = soma( parcela1, parcela2 );
22
23     // Saída de dados/resultados
24     printf("%d + %d = %d\n", parcela1, parcela2, total );
25
26     return 0;
27 }
```

Declaração

Execução

função

Todas as funções precisam ser **declaradas** antes dos pontos de 'chamada' (**execução**).

Funções de associação de resistores

```
main.c
1 // Declaração de bibliotecas
2 #include <stdio.h>
3 // Declaração das funções
4 int serie( int rx, int ry )
5 {
6     return(rx + ry);
7 }
8 int paralelo( int rx, int ry )
9 {
10     int prod;
11     int soma;
12     prod = rx * ry;
13     soma = rx + ry;
14     return( prod / soma );
15 }
16
17 // Programa principal
18 int main()
19 {
20     // Declaração de variáveis
21     int resistor1;
22     int resistor2;
23     int reqSerie;
24     int reqParalelo;
25
26     // Entrada de dados
27     printf("Digite o valor de dois resistores[em ohms]:");
28     scanf("%d %d", &resistor1, &resistor2 );
29
30     // Execução
31     reqSerie = serie( resistor1, resistor2 );
32     reqParalelo = paralelo( resistor1, resistor2 );
33
34     // Saída de dados/resultados
35     printf("%d + %d = %d\n", resistor1, resistor2, reqSerie );
36     printf("%d // %d = %d\n", resistor1, resistor2, reqParalelo );
37
38     return 0;
39 }
```

Declaração

função

Todas as funções precisam ser **declaradas** antes dos pontos de 'chamada' (**execução**).

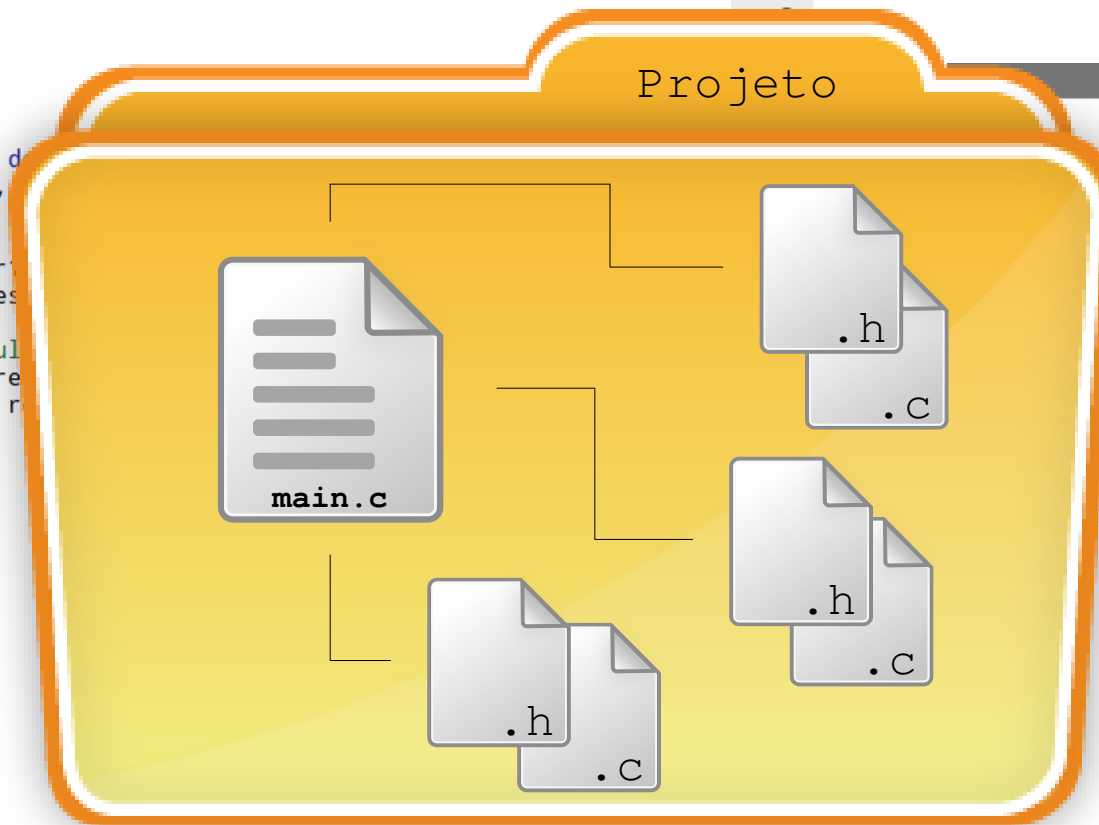
Execução

Organização do Projeto

```
main.c  req.c  :  req.h  :  
1 // Declaração de bibliotecas  
2 #include <stdio.h>  
3 #include "req.h"  
4  
5 // Programa principal  
6 int main()  
7 {  
8     // Declaração de variáveis  
9     int resistor1;  
10    int resistor2;  
11    int reqSerie;  
12    int reqParalelo;  
13  
14    // Entrada de dados  
15    printf("Digite o valor de d  
16    scanf("%d %d", &resistor1,  
17  
18    // Execução  
19    reqSerie = serie( resistor  
20    reqParalelo = paralelo( res  
21  
22    // Saída de dados/resul  
23    printf("%d + %d = %d\n", re  
24    printf("%d // %d = %d\n", r  
25  
26    return 0;  
27 }
```

```
main.c  req.c  :  req.h  :  
1 #ifndef REQ_H  
2 #define REQ_H  
3  
4 int serie( int rx, int ry );  
5 int paralelo( int rx, int ry );  
6  
7 #endif
```

```
req.c  :  req.h  :  
// Declaração das funções de  
// calculo para  
// associação de resistores  
serie( int rx, int ry )  
return(rx + ry);  
paralelo( int rx, int ry )  
  
int prod;  
int soma;  
prod = rx * ry;  
soma = rx + ry;  
return( prod / soma );
```



Organização do projeto

The diagram illustrates the organization of a C project with three files: `main.c`, `req.c`, and `req.h`. Arrows indicate the flow of code and dependencies.

main.c (lines 1-27):

```
1 // Declaração de bibliotecas
2 #include <stdio.h>
3 #include "req.h"
4
5 // Programa principal
6 int main()
7 {
8     // Declaração de variáveis
9     int resistor1;
10    int resistor2;
11    int reqSerie;
12    int reqParalelo;
13
14    // Entrada de dados
15    printf("Digite o valor de dois resistores[em ohms]: ");
16    scanf("%d %d", &resistor1, &resistor2 );
17
18    // Execução
19    reqSerie = serie( resistor1, resistor2 );
20    reqParalelo = paralelo( resistor1, resistor2 );
21
22    // Saída de dados/resultados
23    printf("%d + %d = %d\n", resistor1, resistor2, reqSerie );
24    printf("%d // %d = %d\n", resistor1, resistor2, reqParalelo );
25
26    return 0;
27 }
```

req.h (lines 1-7):

```
1 #ifndef REQ_H
2 #define REQ_H
3
4 int serie( int rx, int ry );
5 int paralelo( int rx, int ry );
6
7 #endif
```

req.c (lines 1-17):

```
1 // Declaração das funções de
2 // calculo para
3 // associação de resistores
4
5
6 int serie( int rx, int ry )
7 {
8     return(rx + ry);
9 }
10 int paralelo( int rx, int ry )
11 {
12     int prod;
13     int soma;
14     prod = rx * ry;
15     soma = rx + ry;
16     return( prod / soma );
17 }
```

Arrows indicate the following dependencies and flow:

- An arrow from `main.c` line 3 to `req.h` line 1, indicating the inclusion of the header file.
- An arrow from `main.c` line 19 to `req.c` line 6, indicating the call to the `serie` function.
- An arrow from `main.c` line 20 to `req.c` line 10, indicating the call to the `paralelo` function.

Fluxograma

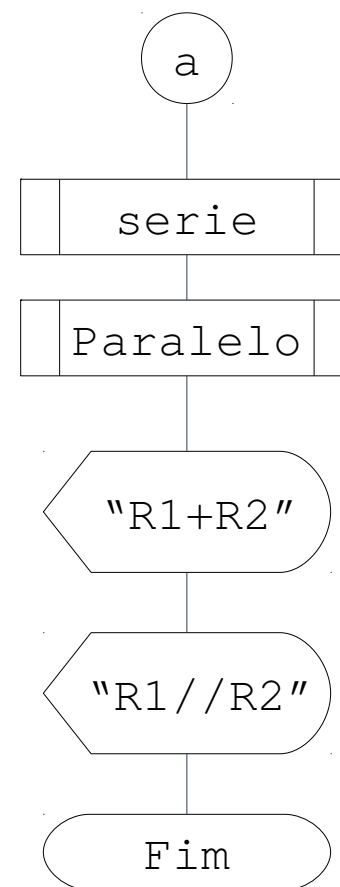
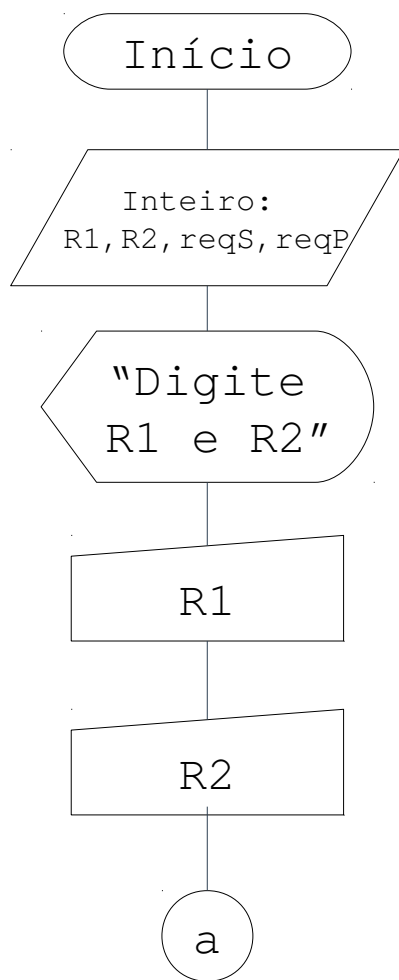
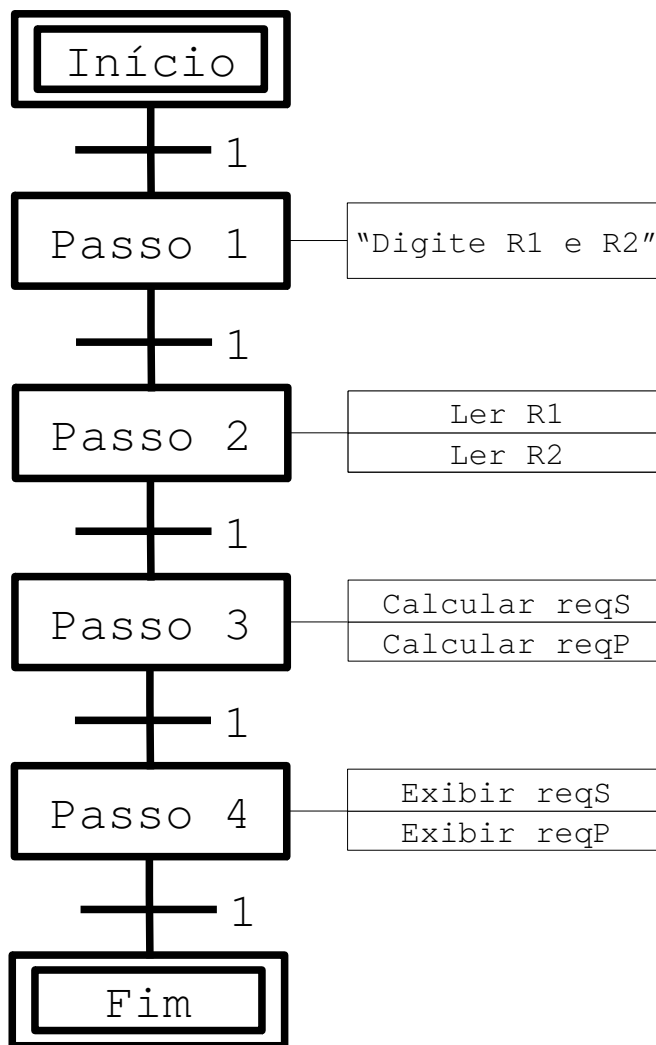
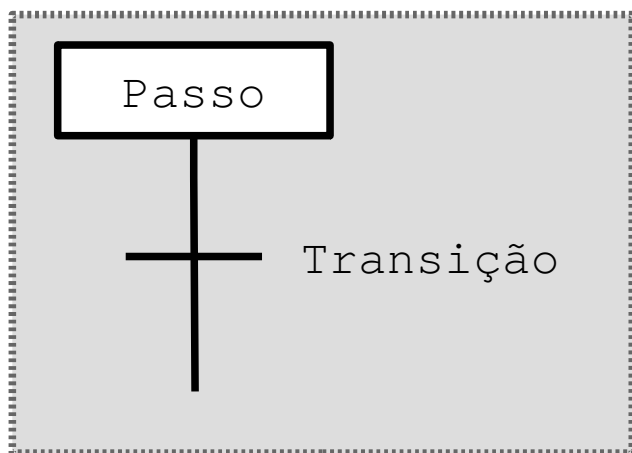


Grafico de Funções Sequenciais

SFC – Sequential Function Chart





Site: josewrpereira.github.io/ddp/
E-mail: josewrpereira@outlook.com
jose.wpereira@senaisp.edu.br