

Funções();

**Problema: Em
uma agenda:**

- 1. Imprima todos**
- 2. Imprima
apenas um**

**Imprimir
Todos**

```
switch(opcao) {  
    case 1:  
        for(int i = 0; i < n; ++i) {  
            printf("Nome: %s\n", nome[i]);  
            printf("Idade: %d\n", idade[i]);  
            printf("Telefone: %s\n", telefone[i]);  
        }  
        break;  
    case 2:  
        int id = 0;  
        scanf("%d", &id);  
        printf("Nome: %s\n", nome[id]);  
        printf("Idade: %d\n", idade[id]);  
        printf("Telefone: %s\n", telefone[id]);  
        break;  
}
```

**Imprimir
UM**

Imprimir
Todos

```
switch(opcao) {  
    case 1:  
        for(int i = 0; i < n; ++i) {  
            printf("Nome: %s\n", nome[i]);  
            printf("Idade: %d\n", idade[i]);  
            printf("Telefone: %s\n", telefone[i]);  
        }  
        break;  
    case 2:  
        int id = 0;  
        scanf("%d", &id);  
        printf("Nome: %s\n", nome[id]);  
        printf("Idade: %d\n", idade[id]);  
        printf("Telefone: %s\n", telefone[id]);  
        break;  
}
```

Imprimir
UM

Imprimir
Todos

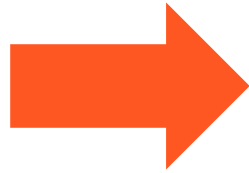
```
switch(opcao) {  
    case 1:  
        for(int i = 0; i < n; ++i) {  
            printf("Nome: %s\n", nome[i]);  
            printf("Idade: %d\n", idade[i]);  
            printf("Telefone: %s\n", telefone[i]);  
        }  
        break;  
    case 2:  
        int id = 0;  
        scanf("%d", &id);  
        printf("Nome: %s\n", nome[id]);  
        printf("Idade: %d\n", idade[id]);  
        printf("Telefone: %s\n", telefone[id]);  
        break;  
}
```

Imprimir
UM

São IGUAIS!

```
printf("Nome: %s\n", nome[i]);  
printf("Idade: %d\n", idade[i]);  
printf("Telefone: %s\n", telefone[i]);
```

Imprimir
Pessoa



```
printf("Nome: %s\n", nome[i]);  
printf("Idade: %d\n", idade[i]);  
printf("Telefone: %s\n", telefone[i]);
```

Imprimir
Todos

```
switch(opcao) {  
    case 1:  
        for(int i = 0; i < n; ++i) {  
            printf("Nome: %s\n", nome[i]);  
            printf("Idade: %d\n", idade[i]);  
            printf("Telefone: %s\n", telefone[i]);  
        }  
        break;  
    case 2:  
        int id = 0;  
        scanf("%d", &id);  
        printf("Nome: %s\n", nome[id]);  
        printf("Idade: %d\n", idade[id]);  
        printf("Telefone: %s\n", telefone[id]);  
        break;  
}
```

Imprimir
UM


```
switch(opcao) {  
    case 1:  
        for(int i = 0; i < n; ++i) {  
            }  
        break;  
    case 2:  
        int id = 0;  
        scanf("%d", &id);  
        }  
}
```

Imprimir
Todos

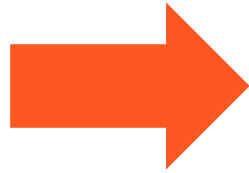
Imprimir
Pessoa

Imprimir
UM

Imprimir
Pessoa

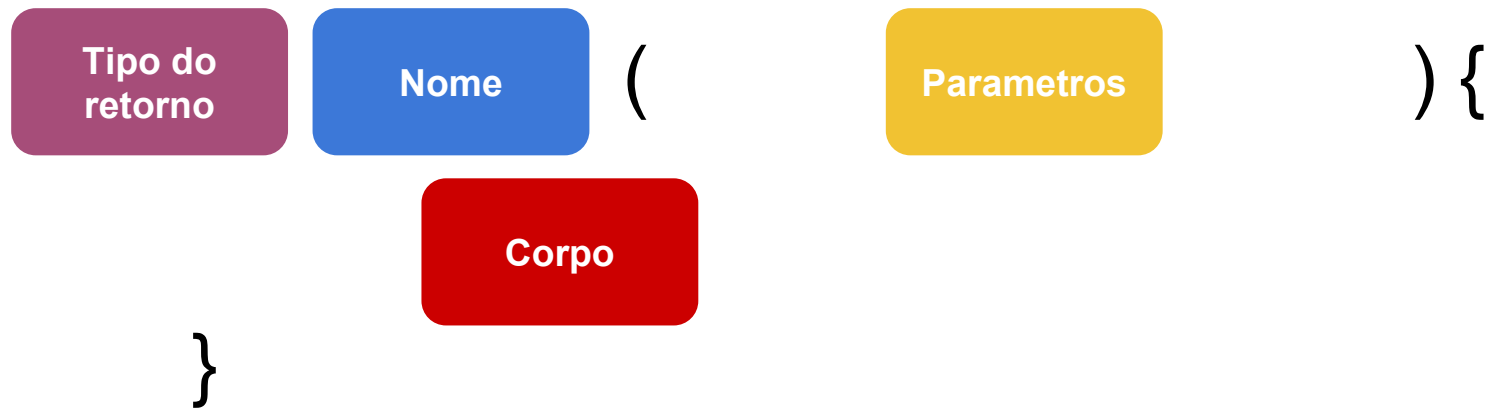
**Função é: Parte
do código que
pode ser
reutilizado**

Imprimir
Pessoa



```
printf("Nome: %s\n", nome[i]);  
printf("Idade: %d\n", idade[i]);  
printf("Telefone: %s\n", telefone[i]);
```

Syntax



Syntax

int, char, double, float



Tipo do
retorno

Nome

(

Parametros

) {

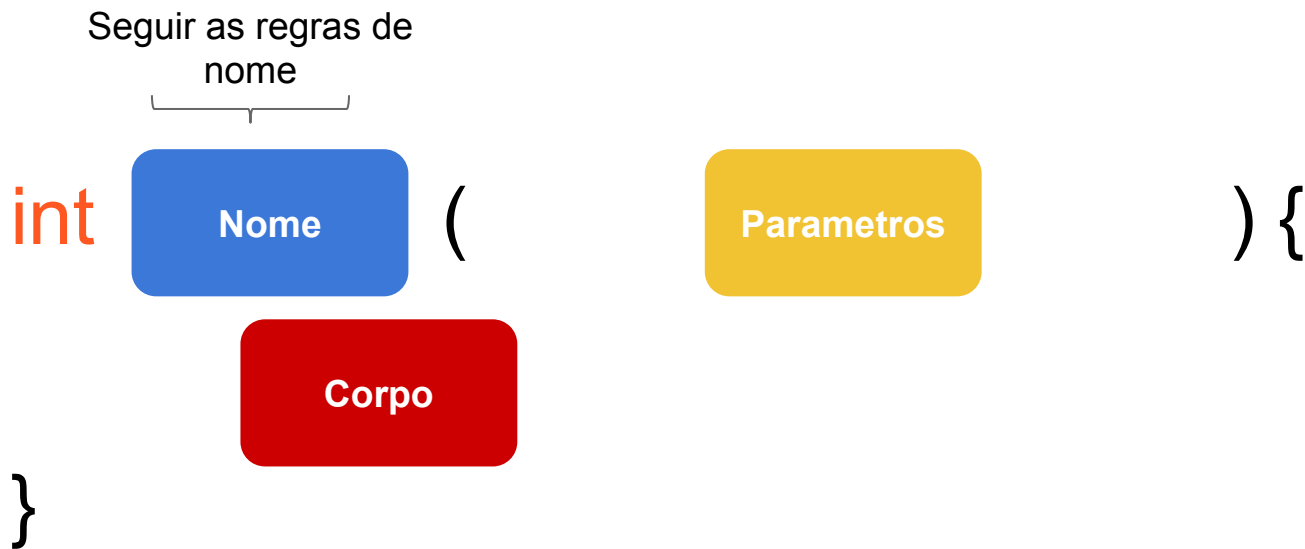
Corpo

}

Syntax

```
int Nome ( Parametros ) {  
    Corpo  
}
```

Syntax



Syntax

Seguir as regras de
nome



int potencia (

Parametros

) {

Corpo

}

Syntax

Variáveis que serão
utilizadas dentro da função

int potencia (



Parametros

) {

Corpo

}

Syntax

Mesma syntax de
declaração de variáveis

Variáveis que serão
utilizadas dentro da função

int potencia (

Parametros

) {

Corpo

}

Syntax

Mesma syntax de
declaração de variáveis

Variáveis que serão
utilizadas dentro da função



```
int potencia (int base, int expoente) {
```

Corpo

```
}
```

Syntax

Mesma syntax de
declaração de variáveis

Variáveis que serão
utilizadas dentro da função

Quantas você
precisar! Separadas
por vírgula



```
int potencia (int base, int expoente) {
```

Corpo

```
}
```

Syntax

```
int potencia (int base, int expoente) {
```

Corpo

Qualquer código: loops,
condicionais, variáveis,
arrays, ...

```
}
```

Syntax

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

A verdade sobre o return

return

Retornar um
valor

—


```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

```
➡ int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

A função é definida
antes da main

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```



```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Início da execução
do programa,
sempre é na main

```
➡ int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Ler o número



```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

```
int main() {  
    int numero;  
    scanf("%d", &numero);
```



```
    int p = potencia(numero, 5);  
    printf("%d\n", p);
```

Chamada da função
potência

```
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }
```

```
    return 0;  
}
```

➡ `int potencia (int base, int expoente) {`
 `...`
 `return numero_elevado;`
`}`

Início da execução
da função

`int main() {`
 `int numero;`
 `scanf("%d", &numero);`


base = numero

➡ `int p = potencia(numero, 5);`
 `printf("%d\n", p);`


expoente = 5

`for(int i = 0; i < p; i++) {`
 `printf("%d ", i);`
 `}`

`return 0;`
`}`



```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```



```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Retorno do resultado
da função

```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```



```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Retorno do resultado
da função

```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

p = Resultado
que a função
retornou

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Imprime p

```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```



```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}
```

Continua a executar o
programa

```
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```



```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}  
  
int main() {
```

Moral da história

```
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}  
  
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

Argumentos
assumem os
valores passados

Função foi
chamada

Função
retornou

```
int potencia (int base, int expoente) {  
    ...  
    return numero_elevado;  
}  
  
int main() {  
    int numero;  
    scanf("%d", &numero);  
  
    int p = potencia(numero, 5);  
    printf("%d\n", p);  
  
    for(int i = 0; i < p; i++) {  
        printf("%d ", i);  
    }  
  
    return 0;  
}
```

Argumentos
assumem os
valores passados

Função foi
chamada

Volta executar do
ponto onde foi
chamada

Vantagens de usar funções

1. Evita códigos repetidos
2. Encurta o tamanho do seu código
3. Melhora a manutenção do seu código
4. Deixa o código mais legível

**Testem: faça e
use uma função
que some dois
números**

FAQ - Perguntas Mais Frequentes

Qual o máximo de argumentos eu posso ter?

```
int funcao (int a) {  
    ...  
    return n;  
}
```

Infinitos (até onde o PC aguentar)

Qual o máximo de funções eu posso ter?

```
int funcao1 (int a) {  
    ...  
    return n;  
}  
char funcao2 (int a) {  
    ...  
    return n;  
}  
double funcao3 (int a) {  
    ...  
    return n;  
}
```

Infinitas (até onde o PC aguentar)

Posso ter argumentos de tipos misturados?

```
char funcao (int a, char b, double c) {  
    ...  
    return letra;  
}
```

Sim, pode misturar os tipos

Posso retornar um valor fixo na minha função?

```
double pi () {  
    ...  
    return 3.141516;  
}
```

Sim!

Posso falar que meu retorno é int e retornar um char?

```
int soma2 (int n) {  
    ...  
    return 'a';  
}
```



Não! O tipo do retorno tem que ser o mesmo tipo que vai ser retornado

Posso ter uma função chamada x e uma variável chamada x?

```
int x (int n) {  
    ...  
}  
...  
int x = 0;
```



Não! O nome da função deve ser único.

E se eu não quiser retornar NADA, o que eu faço?

void o tipo
segreto

Vazio
Nada

E se eu não quiser retornar NADA, o que eu faço?

```
void imprima_pessoa (int id) {  
    printf("Pessoa de id = %d\n", id);  
}
```

Use o tipo void como retorno

Posso usar o return em uma função do tipo void?

```
void imprima_pessoa (int id) {  
    if (id < 0) {  
        return;  
    }  
    printf("Pessoa de id = %d\n", id);  
}
```

Sim, mas o return deve ser vazio

**Testem: faça uma
função do tipo void,
para imprimir os
números de 1 a N**

return;

