Passagem de Argumentos por Referência

Muitas vezes, temos funções que definem valores que serão usados fora destas funções. Nestes casos, para usarmos tais valores, precisávamos retorná-los e atribuí-los a uma variável.

Como exemplo, temos a função de calcular idade. Nesta função, a idade é calculada e seu valor precisa ser usado em outra função. Então, a função de calcular idade retorna o valor (Exemplo A) e este valor é atribuído a uma variável, onde a função do cálculo foi originalmente chamada (Exemplo B).

Exemplo A	**Exemplo B**

int calcularIdade(int ano){
int idade;
idade = 2019-ano;
return idade;
}
	int main(int argc, char** argv) {
	int idade;
	int ano;
	
	printf("\nDigite seu ano de nascimento:");
	scanf("%d",&ano);
	
	idade=calcularIdade(ano);
	
	printf("\nVoce tem mais ou menos %d anos",idade);
	return 0;
	}

Há, porém, uma segunda forma de atribuir valores a variáveis sem a necessidade da função retornar estes valores. Lembre-se que, ao usarmos ponteiros, temos o seguinte recurso:

```
1 int a;     //Declaração de uma variável int
2 int *b;    //Declaração de uma variável ponteiro para int (esta variável armazena endereços de variáveis inteiras)
3 b=&a;      //A variável ponteiro agora armazena o endereço da variável a
4 *b=10;     //Usando a notação * junto com o ponteiro, estamos atribuindo o valor 10 à variável a (porque b armazena
             o endereço de a)
```

Em resumo: na linha 4 temos um recurso que permite atribuir uma valor a uma variável usando o ponteiro para esta variável (no exemplo, o ponteiro b é usado para atribuirmos um valor à variável a).
Este mesmo recurso pode ser usado entre funções. Usando o Exemplo A e B, podemos adaptá-los da seguinte forma:

Exemplo A	**Exemplo B**

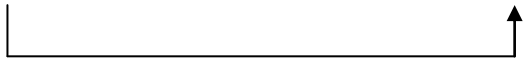
void calcularIdade(int ano, int * idade){
*idade = 2019-ano;
}
	int main(int argc, char** argv) {
	int idade;
	int ano;
	printf("\nDigite seu ano de nascimento:");
	scanf("%d",&ano);
	
	calcularIdade(ano,&idade);
	
	printf("\nVoce tem mais ou menos %d anos",idade);
	return 0;
	}

Ao olharmos de perto, temos as seguintes mudanças:

```
idade=calcularIdade(ano);   ->   calcularIdade(ano,&idade);
```

Veja que, se antes a função retornava um valor e este valor era atribuído à variável idade, agora passamos o endereço da idade.

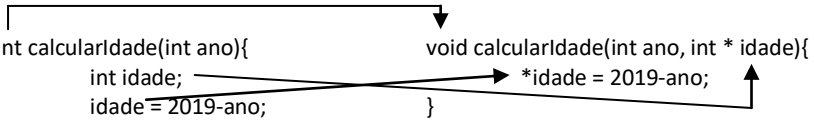
```
idade=calcularIdade(ano);   ->   calcularIdade(ano,&idade);
```



Em relação às mudanças na função calcular idade, na nova função precisaremos criar um parâmetro para receber o endereço passado e não precisamos mais retornar valores, pois estaremos usando o ponteiro para atribuir o valor calculado diretamente para o endereço da variável idade. Deste modo:

```
int calcularIdade(int ano){                    void calcularIdade(int ano, int * idade){
        int idade;                  ->                 *idade = 2019-ano;
        idade = 2019-ano;                      }
        return idade;
}
```

E as mudanças estão marcadas como segue:



```
int calcularIdade(int ano){                    void calcularIdade(int ano, int * idade){
        int idade;                                     *idade = 2019-ano;
        idade = 2019-ano;                      }
        return idade;
}
```

Exercícios sobre Passagem por Referência

1- Altere as chamadas de funções abaixo de forma que a função chamada não tenha retorno e, ao mesmo tempo, a variável que antes receberia o valor retornado possa ser alterado diretamente na função chamada.

```
ano = lerDado();
int ano;
lerDado(&ano);

a = recebendoUmValor();
int a;
recebendoUmValor(&a);

idade = calcularIdade(ano);
int idade, ano;
calcularIdade(ano, &idade);

b = calcularAlgo(a);
int b,a;
calcularAlgo(a, &b);

c = somar(a,b);
int a,b,c;
somar(a,b,&c);

c = fazAlgo(a,b);
int a,b,c;
fazAlgo(a,b,&c);

d = getRA();
int d;
getRA(&d);

e = getValor();
int e;
getValor(&e);
```

2- Altere as funções abaixo, de forma que não retornem mais valores e, ao mesmo tempo, atribuam o valor definido diretamente no endereço de memória recebido.

int retornandoUmNumero(){
a) int a;
a = 40;
return a;
}
	int retornaValorAleatorio(){
	b) int valor;
	srand(time(NULL));
	valor = rand()%10;
	return valor;
	}

int calcularIdade(int ano){
c) int idade;
idade = 2019-ano;
return idade;
}
	int calcularAno(int idade){
	d) int ano;
	ano = 2019-idade;
	return ano;
	}

int somar(int a, int b){
e) int res;
res=a+b;
return res;
}
	int subtrair(int a, int b){
	f) int res;
	res=a-b;
	return res;
	}

a)

```
void retornandoUmNumero(int *a){
*a = 40;
}
```

b)

```
void retornaValorAleatorio(int * valor){
srand(time(NULL));
*valor = rand()%10;
}
```

c)

```
void calcularIdade(int ano, int *idade){
*idade = 2019 - ano;
}
```

d)

```
void calcularAno(int idade, int *ano){
*ano = 2019 - idade;
}
```

e)

```
void somar(int a, int b, int *res){
*res = a + b;
}
```

f)

```
void subtrair(int a, int b, int *res){
*res = a - b;
}
```

3 – Altere os códigos abaixo de forma que eles não retornem mais valores. No lugar, use ponteiros.

```
int main(int argc, char** argv) {
        int valor;
        int quadrado;

        printf("\nDigite um valor:");
        scanf("%d",&valor);

        quadrado=calcularQuadrado(valor);
        calcularQuadrado(valor, &quadrado);
        printf("\n%d ao quadrado = %d ",valor,quadrado);

        return 0;
}
```

```
int calcularQuadrado(int val){
        int quad;
        quad = val * val;
        return quad;
}
 void calcularQuadrado(int val, int *quadrado){
  *quadrado = val * val;
 }
```

```
int main(int argc, char** argv) {
        float celsius;
        float kelvin;

        printf("\nDigite a temperatura em Celsius:");
        scanf("%f",&celsius);

        kelvin=converterCparaK(celsius);
        converterCparaK(celsius, &kelvin);
        printf("\n%.2f Celsius = %.2f Kelvin",celsius,kelvin);

        return 0;
}
```

```
float converterCparaK(float c){
        float k;
        k = c + 273.15f;
        return k;
}
void converterCparaK(float c, float * k){
 *k = c + 273.15f;
}
```

```
int main(int argc, char** argv) {
        float celsius;
        float kelvin;

        printf("\nDigite a temperatura em Celsius:");
        scanf("%f",&celsius);

        kelvin=converterCparaK(celsius);

        printf("\n%.2f Celsius = %.2f Kelvin",celsius,kelvin);

        return 0;
}
```

```
float converterCparaK(float c){
        float k;
        k = c + 273.15f;
        return k;
}
```

```
int main(int argc, char** argv) {

        int valor1, valor2;
        int resultado;


        printf("\nDigite um valor:");
        scanf("%d",&valor1);

        printf("\nDigite outro valor:");
        scanf("%d",&valor2);

        resultado=multiplicar(valor1, valor2);
        multiplicar(valor1, valor2, &resultado);
        printf("\n%d x %d = %d ",valor1, valor2, resultado);

        return 0;
}
```

```
int multiplicar(int a, int b){
        int res;
        res = a * b;
        return res;
}
 void multiplicar(int a, int b, int *resultado){
  *resultado = a*b;
 }
```

4 – Para cada código do exercício 3, preencha a tabela de endereços, supondo que a primeira variável seja sempre alocada no endereço 55FF20

A)

Identificador	Endereço de memória	Valor
valor	00FF20	10
quadrado	00FF1C	100
val	00FF18	10
quadrado	00FF10	00FF1C

B)

Identificador	Endereço de memória	Valor
celsius	00FF20	30
kelvin	00FF1C	303,15
c	00FF18	30
k	00FF10	00FF1C

C)

Identificador	Endereço de memória	Valor
valor1	00FF20	2
valor2	00FF1C	3
resultado	00FF18	6
a	00FF14	2
b	00FF10	3
resultado	00FF08	00FF18