

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
<pre>int calcularIdade(int ano){ int idade; idade = 2019-ano; return idade; }</pre>	<pre>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; }</pre>

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 atribuímos 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:

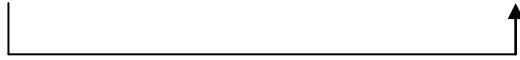
Exemplo A	Exemplo B
<pre>void calcularIdade(int ano, int * idade){ *idade = 2019-ano; }</pre>	<pre>int main(int argc, char** argv) { int idade; int ano; printf("\nDigite seu ano de nascimento:"); scanf("%d",&ano); calcularIdade(ano,&idade); printf("\nVoce tem mais ou menos %d anos",idade); return 0;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

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:

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

E as mudanças estão marcadas como segue:

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

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();

a = recebendoUmValor();

idade = calcularIdade(ano);

b = calcularAlgo(a);

c = somar(a,b);

c = fazAlgo(a,b);

d = getRA();

e = getValor();

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

<pre>int retornandoUmNumero(){ int a; a = 40; return a; }</pre>	<pre>int retornaValorAleatorio(){ int valor; srand(time(NULL)); valor = rand()%10; return valor; }</pre>
<pre>int calcularIdade(int ano){ int idade; idade = 2019-ano; return idade; }</pre>	<pre>int calcularAno(int idade){ int ano; ano = 2019-idade; return ano; }</pre>
<pre>int somar(int a, int b){ int res; res=a+b; return res; }</pre>	<pre>int subtrair(int a, int b){ int res; res=a-b; return res; }</pre>

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

<pre>int main(int argc, char** argv) { int valor; int quadrado; printf("\nDigite um valor:"); scanf("%d",&valor); quadrado=calcularQuadrado(valor); printf("\n%d ao quadrado = %d ",valor,quadrado); return 0; }</pre>	<pre>int calcularQuadrado(int val){ int quad; quad = val * val; return quad; }</pre>
--	--

<pre>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; }</pre>	<pre>float converterCparaK(float c){ float k; k = c + 273.15f; return k; }</pre>
---	--

<pre>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; }</pre>	<pre>float converterCparaK(float c){ float k; k = c + 273.15f; return k; }</pre>
---	--

<pre>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); printf("\n%d x %d = %d ",valor1, valor2, resultado); return 0; }</pre>	<pre>int multiplicar(int a, int b){ int res; res = a * b; return res; }</pre>
--	---

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