

# 1 - SOBRE PONTEIROS

1 - Vamos observar um código

```
int main() { //inicio_main

    printf("\n");

    int contador = 10;

    printf("Antes de incrementar.\n", contador);
    printf("O contador vale :%d\n", contador);

    //chamada de função
    incrementa(contador); //passa a variavel contador para a função incrementa.

    printf("Depois de incrementar.\n", contador);
    printf("O contador vale :%d\n", contador);

    //declarando variaveis
    //entrada_dados
    //processamento_dados
    //saida_dados
    printf("\n");
    printf("\n");
    return 0;
} //fin_main
```

```
//////////////////////////////////FUNÇÕES//////////////////////////////////

//1//
void incrementa(int valor){
    printf("O Antes de incrementar.\n");
    printf("O contador vale :%d\n", valor);

    printf("O Depois de incrementar.\n");
    //valor++;
    printf("O contador vale:%d\n", ++valor);
    //printf("O contador vale :%d\n", valor);
}
```

IMPRIME CONTADOR

~2 FORMAS~  
1 - valor++(incrementa DEPOIS da execução)  
2 - ++valor(incrementa ANTES da execução)

```
Antes de incrementar.
O contador vale :10
O Antes de incrementar.
O contador vale :10
O Depois de incrementar.
O contador vale:11
Depois de incrementar.
O contador vale :10
```

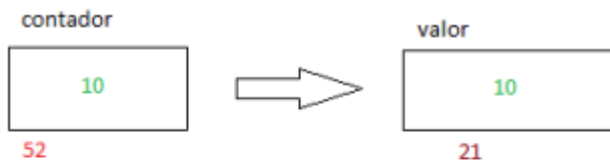
- A QUESTÃO É %, PQ DEPOIS DE **INCREMENTAR** AINDA CONTINUA DANDO 10?

- O que queremos é que quando a função incrementa for chamada, ela adicione 1 ao contador e depois retorne ela incrementada, mas isso não está acontecendo, ela só está incrementando dentro da execução da função.

- Isso acontece pq quando inicializamos **contador = 10;** e passamos **contador** como parametro estamos passando para a função somente o valor 10 = COPIA POR VALOR.

À À À - Quando declaramos uma variável a linguagem C aloca um espaço em memória para colocar este valor.

À



-valor é uma variavel criada dentro da função incrementa, logo ela só existe lá dentro. Quando chamamos a função e passamos o contador como parametro, estamos dando a valor o valor da variavel criada contador, fazendo a PASSAGEM POR COPIA.

Â

- Quando falamos em ponteiros, estamos falando em manipulação de memória.
- `int contador = 10;` esta criando um espaço de memória para ser colocado o valor da variavel contador.
- Â Â Â - Com ponteiro fazemos a manipulação desse espaço de memória.Â Â

- Vamos imprimir o endereço de memória da variavel contador:

```

Antes de incrementar.      1
0 contador vale :10
0 endereço de memória eh: 6422300
0 Antes de incrementar.
0 contador vale :10
0 endereço de memória eh: 6422272
0 Depois de incrementar.
0 contador vale:11
0 endereço de memória eh: 6422272
Depois de incrementar.
0 contador vale :10
0 endereço de memória eh: 6422300
  
```

Â

- Vemos que temos 2 endereços de memória:
- Â Â Â 1 - Endereço onde a variavel contador criada no main esta localizada.
- Â Â Â 2 - Endereço onde a variavel contador, criada na função incrementa esta localizada.

- Esta provando q as variaveis são diferentes, estão em endereços de memória diferentes.

Â Â Â - Passagem por copia de valor.

- Para que o codigo funcione na forma que queremos, ou seja, incrementar o contador, temos que passar o endereço da variavel contador para a função incrementa. E , na propria função, informar que vamos receber um endereço de memória com um \*.

```
incrementa(&contador)
```

```
void incrementa (int *contador){}
```

& -> fornece o endereço de memória da variavel.

```

void incrementa(int *valor); // f

int main(){ //inicio_main

    printf("\n");

    int contador = 10;

    printf("Antes de incrementar
    printf("O contador vale :%d\
    printf("O endereço de memori

    incrementa(&contador); //pas

```

Â

```

////////////////////////////////////////FUNÇÕES////////////////////////////////////////

//1//
void incrementa(int *valor){
    printf("O Antes de incrementar.\n");
    printf("O contador vale :%d\n", valor);
    printf("O endereço de memoria eh: %d\n", &valor);

    printf("O Depois de incrementar.\n");
    //valor++;
    printf("O contador vale:%d\n", ++valor);
    //printf("O contador vale :%d\n", valor);
    printf("O endereço de memoria eh: %d\n", &valor);
}

```

- Para ter acesso ao valor de uma variavel ponteiro colocamos o \*varivel, na frente da variavel.

Â

- Pequeno porem, se imprimirmos como esta descrito na imagem acima, o valor que o contador apresentara serÃ; o do endereÃço de memoria, para apresentar o valor da variavel contador, temos que colocar um \*...

Â Â Â - \*valorÂ = indica que queremos mostrar o valor da variavel ponteiro e nÃ£o seu endereÃço.

- Queremos incrementar tbm esse valor, enÃ£o na linha d baixo fazemos o mesmo processo com o \*..

```

//////////////////////////////////FUNÇÕES//////////////////////////////////

//1//
void incrementa(int *valor){
    printf("0 Antes de incrementar.\n");
    printf("0 contador vale :%d\n", *valor);
    printf("0 endereço de memoria eh: %d\n",&valor);

    printf("0 Depois de incrementar.\n");
    //valor++;
    printf("0 contador vale:%d\n",++(*valor));
    //printf("0 contador vale :%d\n",valor);
    printf("0 endereço de memoria eh: %d\n",&valor);
}

```

Â Â

```

Antes de incrementar.
0 contador vale :10
0 endereço de memoria eh: 6422300
0 Antes de incrementar.
0 contador vale :10
0 endereço de memoria eh: 6422300
0 Depois de incrementar.
0 contador vale:11
0 endereço de memoria eh: 6422300
Depois de incrementar.
0 contador vale :11
0 endereço de memoria eh: 6422300

```

Â Â

- Vemos que os endereços das variáveis ainda são diferentes, o que fizemos foi uma PASSAGEM DE VALOR POR REFERENCIA...Onde damos a função o endereço de memória da variável, e assim a função em vez de criar outra variável, acessa o endereço e altera a variável original.

```

int main(){//inicio_main
    printf("\n");

    int i, j, *pi, **ppj;

    i = 7;
    j = 10;

    printf("Os valores de i e j sao: %d, %d\n",i,j);
    printf("Os end. de memoria de i e j sao: %d, %d\n",&i,&j);

    pi = &i;
    ppj = &j;

    printf("Os valores de pi e pj sao: %x, %x\n",pi,pj);
    printf("Os end. de memoria de pi e pj sao: %d, %d\n",&pi,&ppj);
    printf("%d %d",*pi,*ppj);

    *pi = 20;
    *pj = 15;
    printf("Os valores de i e j sao: %d, %d\n",i,j);

    *pj = *pi;
    printf("Os valores de i e j sao: %d, %d\n",i,j);

    pi = pj;
    printf("Os valores de pi e pj sao: %d, %d\n",pi,pj);

    ppj = &ppj;
    **ppj = 70;
}

```

#### MEMORIA DO COMPUTADOR

*pi	*pj	**ppj
315 72	72	120
100	120	250
i	j	
7 20	70 10 15 20	
315	72	

Â