1 - SOBRE PONTEIROS

1 - Vamos observar um codigo

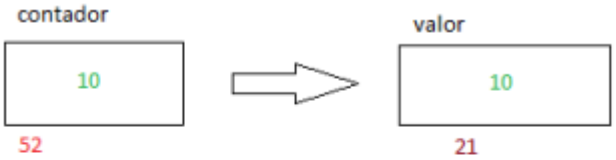
```c
int main(){//inicio_main

    printf("\n");

    int contador = 10;

    printf("Antes de incrementar.\n", contador);
    printf("O contador vale :%d\n",contador);
    chamada de função 1
    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      You, a few seconds ago • Uncommitted changes
    //processamento_dados
    //saida_dados
    printf("\n");
    printf("\n");
    return 0;
}//fin_main
```

```c
////////////////////////////////////////FUNÇÕES/////////////////////////////////////////////////

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

    printf("O Depois de incrementar.\n");
    //valor++;                                  ~2 FORMAS~
    printf("O contador vale:%d\n",++valor);     1 - valor++(incrementa DEPOIS da execução)
    //printf("O contador vale :%d\n",valor);    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 QUESTÃƒO Ã‰, PQ DEPOIS DE **INCREMENTAR** AINDA CONTINUA DANDO 10?
- O que queriamos Ã© que quando a funÃ§Ã£o incrementa for chamada, ela adicione 1 ao contador e depois retorne ela incrementada, mas isso nÃ£o esta acontecendo, ela so esta incrementando dentro da execuÃ§Ã£o da funÃ§Ã£o.

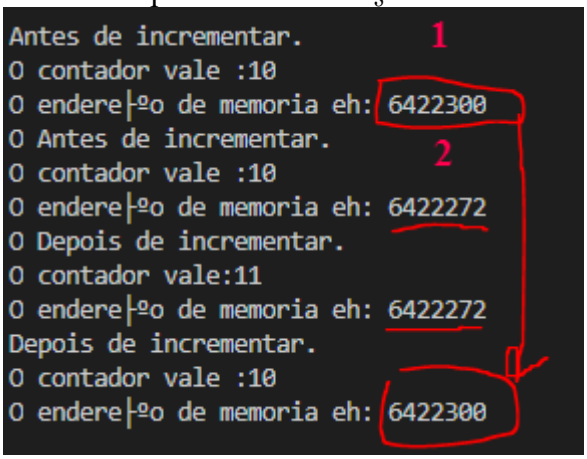
- Isso acontece pq quando inicializamos **contador =10;** e passamos **contador** como parametro estamos passando para a funÃ§Ã£o somente o valor 10 = COPIA POR VALOR.
Â Â Â - Quando declaramos uma variavel a linguagem C aloca um espaÃ§o em memoria para colocar este valor.
Â



-valor é uma variavel criada dentro da função incrementa, logo ela so existe la 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 manipulaÃ§Ã£o de memoria.
- int contador =10; esta criando um espaÃ§o de memoria para ser colocado o valor da variavel contador.
Â Â Â - Com ponteiro fazemos a manipulaÃ§Ã£o desse espaÃ§o de memoria.Â Â

- Vamos imprimir o endereÃ§o de memoria da varivel contador:



Â
- Vemos que temos 2 endereÃ§os de memoria:
Â Â Â 1 - EndereÃ§o onde a variavel contador criada no main esta localizada.
Â Â Â 2 - EndereÃ§o onde a variavel contador, criada na funÃ§Ã£o incrementa esta localizada.

- Esta provando q as variaveis sÃ£o diferentes, estÃ£o em endereÃ§os de memoria 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 endereÃ§o da variavel contador para a funÃ§Ã£o incrementa. E , na propria funÃ§Ã£o, informar que vamos receber um endereÃ§o de memoria com um *.

```
incrementa(&contador)

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

& -> fornece o endereÃ§o de memoria da variavel.

```c
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

```c
////////////////////////////////////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 endereÃ§o de memoria, para apresentar o valor da variavel contador, temos que colocar um *...
Â Â Â Â - *valorÂ = indica que queremos mostrar o valor da variavel ponteiro e nÃ£o seu endereÃ§o.
- Queremos incrementar tbm esse valor, enÃ£o na linha d baixo fazemos o mesmo processo com o *..

```c
////////////////////////////////////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));      You, a few
    //printf("O contador vale :%d\n",valor);
    printf("O endereço de memoria eh: %d\n",&valor);
```

Â Â

```
Antes de incrementar.
O contador vale :10
O endere├ºo de memoria eh: 6422300
O Antes de incrementar.
O contador vale :10
O endere├ºo de memoria eh: 6422300
O Depois de incrementar.
O contador vale:11
O endere├ºo de memoria eh: 6422300
Depois de incrementar.
O contador vale :11
O endere├ºo de memoria eh: 6422300
```

Â Â

- Vemos que os endereÃ§os das variaveis ainda sÃ£o diferentes, o que fizemos foi uma PASSAGEM DE VALOR POR REFERENCIA...Onde damos a funÃ§Ã£o o endereÃ§o de memoria da variavel, e assim a funÃ§Ã£o emv ez de criar outra variavel, acessa o endereÃ§o e altera a variavel original.