

## Conceito (Cap.7)

### Variável

É uma posição (endereço) nomeada da memória, que é usada para guardar um valor que pode ser modificado pelo programa.

Toda vez que “**cont**” é chamado pelo programa, ele se refere ao valor contido no endereço 1004h da memória.

Endereços de memória não utilizados, normalmente carregam um “lixo”.

Nome	End.	Valor
tamanho	1000	0000000A
cont	1004	00000025
maior	1008	0000B001
	100C	02
contptr	100D	00001004
	1011	C4
	.	.
	.	.
	.	.
		<b>Memória</b>

**OBS.:** O endereço e seu valor normalmente são representados em hexadecimal.

## Conceito (Cap.7)

Os **ponteiros** são variáveis que contêm endereços de memória como valores. São usados para fazer referência indireta a um valor.

Nome	End.	Valor
tamanho	1000	0000000A
cont	1004	00000025
maior	1008	0000B001
	100C	02
contptr	100D	00001004
	1011	C4
	.	.
	.	.
	.	.
		<b>Memória</b>

**contptr** faz referência indireta a uma variável cujo valor é 25h.

**cont** faz uma referência direta a uma variável cujo valor é 25h.

## Características dos Ponteiros (Cap.7)

- É uma variável;
- Grande poder x grande responsabilidade;
- Simula chamadas por referência;
- Permite a criação de estruturas de tamanho variável em tempo de execução (como listas encadeadas);

## Declaração (Cap.7)

O ponteiro é uma variável que “aponta” para outra cujo conteúdo é de um “tipo específico”. A declaração é feita da seguinte forma:

**tipo** \*nome;

nome, deve-se usar “ \* ” para informar ao compilador que é um ponteiro.

Tipo do valor que o ponteiro vai referenciar.

**OBS:** Muitos programadores utilizam o sufixo **ptr** para lembrar que a variável é do tipo ponteiro.

## Declaração (Cap.7)

### Exemplos:

```
float *A, *B;
```

Do tipo float

"\*" Indica que A é um ponteiro

Ponteiro

```
int *contptr, cont;
```

do tipo inteiro

variável do tipo inteiro

## Inicialização (Cap.7)

Os ponteiros podem ser inicializados ao serem declarados ou em uma atribuição.

Tipos de inicialização com:

- zero
- endereço
- NULL(stdio.h)

Exemplo: `int * aptr, b;`

```
aptr = 0;
```

```
aptr = NULL;
```

```
int *aptr = NULL;
```

**OBS: Sempre inicialize um ponteiro ao declará-lo (Boa prática)**

## Operadores de Ponteiros (Cap.7)

**Operador &:** devolve o endereço de uma variável.

Exemplo:

```
int y=15;  
int x;  
int *yptr;  
yptr = &y;
```

**yptr** é um ponteiro  
para int

É atribuído à **yptr** o endereço  
de **y**, ou seja, **yptr** passa a  
apontar para **y**.

## Operadores de Ponteiros (Cap.7)

**Operador \*:** devolve o valor da variável apontada pelo operando.

Exemplo:    `x = *yptr;`

Atribui o valor  
apontado por **yptr** a **x**

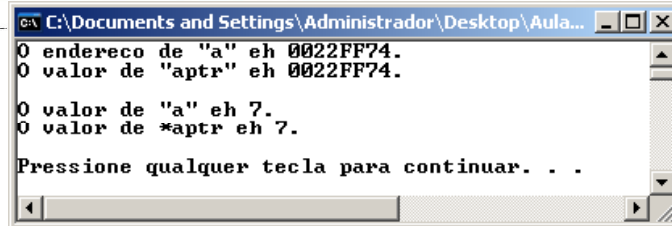
## Operadores de Ponteiros (Cap.7)

### Execução dos Exemplos:

```
int y=15;  
int x;  
int *yptr;  
yptr = &y;  
x = *yptr;
```

Nome	End.	Valor
y	3000	00 00 00 0F
x	3004	00 00 00 0F
yptr	3008	00 00 30 00
:		:
:		:
:		:
Memória		

## Exemplo (Cap.7)



```
main()  
{  
    int a, *aptr;  
  
    a=7;  
    aptr=&a;  
    printf("O endereco de \"a\" eh %p.\n"  
           "O valor de \"aptr\" eh %p.\n\n", &a, aptr);  
    printf("O valor de \"a\" eh %d.\n"  
           "O valor de *aptr eh %d.\n\n", a, *aptr);  
    system("pause");  
}
```

## Chamada de Função por Referência

- **Caso dos *ARRAYS*:** é passado para a função o endereço de início do *array*.
- **Caso dos *ESCALARES*:** é passado o valor da variável.
- **PONTEIRO:** permite que seja passado o endereço onde a variável está armazenada.

## Exemplo: (Cap.7)







### a) Passagem por valores:

```
int cuboporvalor(int);  
  
main()  
{  
    int num=5;  
    printf("O valor original de 'num' eh %d.\n\n", num);  
    num = cuboporvalor(num);  
    printf("O novo valor de 'num' eh %d.\n\n", num);  
    system("pause");  
}  
  
int cuboporvalor(int n)  
{  
    return n*n*n;  
}
```

C:\Documents and Settings\Administrador\Desktop\Aula  
O valor original de 'num' eh 5.  
O novo valor de 'num' eh 125.  
Pressione qualquer tecla para continuar. .

## Exemplo: (Cap.7)

### Análise de uma típica chamada por valor:

<code>int cuboporvalor(int);</code>				
	<code>main()</code>	<b>Nome</b>	<b>End.</b>	<b>Valor</b>
	{	num	3000	00 00 00 7D
	<code>int num=5;</code>			
	<code>num=cuboporvalor(num);</code>	n	3004	00 00 00 05
	<code>...</code>		3008	01 F8
	}		.	.
	<code>int cuboporvalor(int n)</code>		.	.
	{		.	.
	<code>return n*n*n;</code>		Memória	.
	}			

`n*n*n = 125 = 7Dh`

## Exemplo: (Cap.7)

### b) Passagem de referência

```
void cuboporreferencia(int *);
```

```
main()
{
    int num=5;
    printf("O valor original de 'num' eh %d.\n\n", num);
    cuboporreferencia(&num);
    printf("O novo valor de 'num' eh %d.\n\n", num);
    system("pause");
}
```

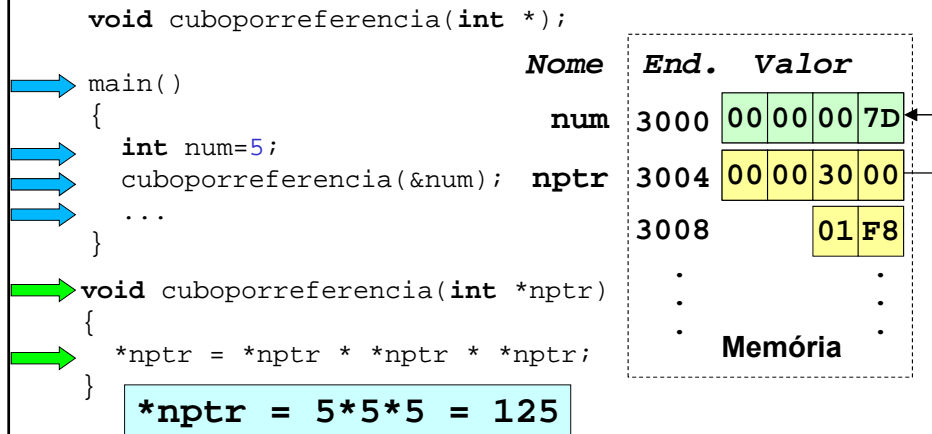
```
void cuboporreferencia(int *nptr)
{
    *nptr = *nptr * *nptr * *nptr;
}
```

O valor original de 'num' eh 5.  
O novo valor de 'num' eh 125.  
Pressione qualquer tecla para continuar. . . .



## Exemplo: (Cap.7)

### Análise de uma típica chamada por referência:



## Qualificador Const. (Cap.7)

O *qualificador* **const** permite ao programador informar ao compilador que o valor de uma determinada variável não deve ser modificado.

**Objetivo:** propiciar um pouco mais de proteção.

## Qualificador Const. (Cap.7)

```
void imprimecaracteres(const char *);
```

```
main()
```

```
{
```

```
    char string[]="Ola pessoal!";
```

```
    printf("O texto eh: ");
```

```
    imprimecaracteres(string);
```

```
    putchar('\n');
```

```
    putchar('\n');
```

```
    system("pause");
```

```
}
```

```
void imprimecaracteres(const char *S)
```

```
{
```

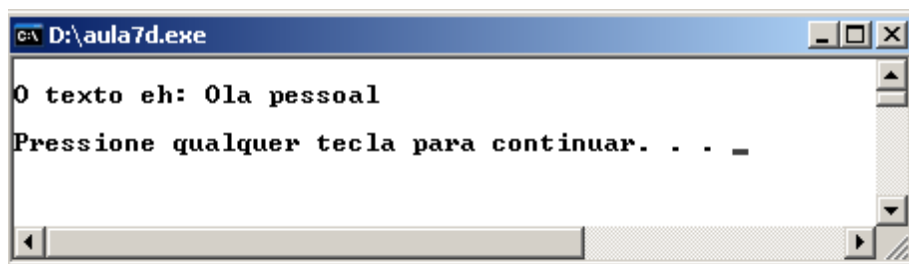
```
    for(; *S!=0; S++)
```

```
        putchar(*S);
```

```
}
```

O valor que será passado não  
pode ser modificado

## Qualificador Const. (Cap.7)



## Aritmética de Ponteiros (Cap.7)

### Tipos de Operações:

- Incremento (+ +)
- Decremento (- -)
- Adição de um inteiro
- Subtração de um inteiro
- Subtração de ponteiros

**OBS:** A aritmética de ponteiros difere da convencional. O incremento de um ponteiro corresponde um deslocamento de tamanho igual tipo da variável em bytes.

## Aritmética do Pontoeiro (Cap.7)

### Exemplo

→ `int *xptr, x;`

→ `xptr = &x;`

→ `xptr++;`

→ `xptr+=2;`

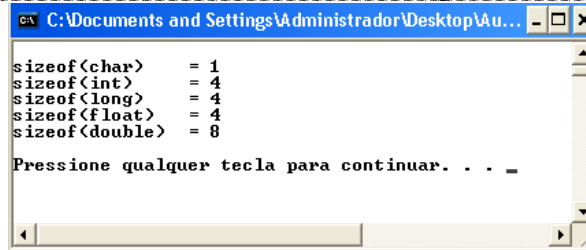
→ `xptr--;`

Nome	End.	Valor
xptr	3000	00 00 30 0C
x	3004	00 00 30 00
	3008	01 F8 E2 FF
	300C	41 14 82 28
	3010	FF 00 3B B3
	.	.
	.	.
	.	.
	Memória	

## Função Sizeof (Cap.7)

Determina o tamanho de um elemento em bytes.

```
main()
{
    system("color f0");
    printf("\nsizeof(char)\t= %d\n", sizeof(char));
    printf("sizeof(int)\t= %d\n", sizeof(int));
    printf("sizeof(long)\t= %d\n", sizeof(long));
    printf("sizeof(float)\t= %d\n", sizeof(float));
    printf("sizeof(double)\t= %d\n\n", sizeof(double));
    system("pause");
}
```



The screenshot shows a command prompt window with the title "C:\Documents and Settings\Administrador\Desktop\Au...". The output of the program is displayed as follows:

```
sizeof(char)      = 1
sizeof(int)       = 4
sizeof(long)      = 4
sizeof(float)     = 4
sizeof(double)    = 8

Pressione qualquer tecla para continuar. . . _
```

## Ponteiros e Arrays (Cap.7)

→ Estão intimamente relacionados.

→ Exemplo:

```
int b[5];
int *bptr;
```

```
bptr = b;
bptr = &b[0];
*(bptr + 3) → b[3];
```

**bptr, b e &b[0] acima são formas diferentes de representar o mesmo endereço.**

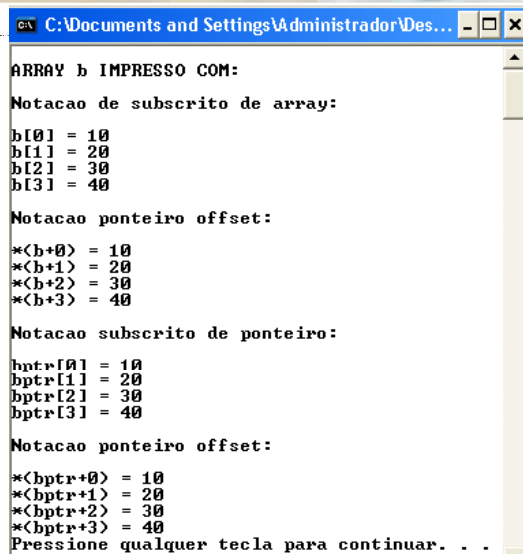
## Programa Exemplo (Cap.4)

```
main()
{
    int i, offset, b[]={10,20,30,40};
    int *bptr=b;

    printf("\nARRAY b IMPRESSO COM:\n\n"
           "Notacao de subscrito de array:\n\n");
    for(i=0; i<=3; i++)
        printf("b[%d] = %d\n", i, b[i]);
    printf("\nNotacao ponteiro offset:\n\n");
    for(offset=0; offset<=3; offset++)
        printf("(b+%d) = %d\n", offset, *(b+offset));
    printf("\nNotacao subscrito de ponteiro:\n\n");
    for(i=0; i<=3; i++)
        printf("bptr[%d] = %d\n", i, bptr[i]);
    printf("\nNotacao ponteiro offset:\n\n");
    for(offset=0; offset<=3; offset++)
        printf("(bptr+%d) = %d\n", offset, *(bptr+offset));

    system("pause");
}
```

## Programa Exemplo (Cap.4)



```
C:\Documents and Settings\Administrador\Des...
ARRAY b IMPRESSO COM:
Notacao de subscrito de array:
b[0] = 10
b[1] = 20
b[2] = 30
b[3] = 40

Notacao ponteiro offset:
*(b+0) = 10
*(b+1) = 20
*(b+2) = 30
*(b+3) = 40

Notacao subscrito de ponteiro:
bptr[0] = 10
bptr[1] = 20
bptr[2] = 30
bptr[3] = 40

Notacao ponteiro offset:
*(bptr+0) = 10
*(bptr+1) = 20
*(bptr+2) = 30
*(bptr+3) = 40
Pressione qualquer tecla para continuar. . .
```

## Exercício (Cap.7)

**Estudar os seguintes exemplos:**

**- Pg 218, fig 7.6 e fig. 7.7**

**- Pg 233, fig 7.20**

## Exercício (Cap.7)

```
int misterio(int, int);

main()
{
    int x, y;

    printf("Entre com dois numeros: ");
    scanf("%d%d", &x,&y);
    printf("\nO resultado eh: %d\n\n", misterio(x,y));
    system("pause");
}

int misterio(int a, int b)
{
    if (b==1) return a;
    else return a+misterio(a,b-1);
}
```