

- Faça o quadro de memória e mostre a saída na tela:

```

int *x1;      int x2;      int *x3;

x1 = (int *) malloc (sizeof(int));
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("\nx1(%p)(%i)(%p) x2(%i)(%p) x3(%p)(%i)(%p)", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

```

Memória	Tela
x1 // *x1 = 20	x1(7Ah)(9B7)(lixo) x2(lixo)(2DE) x3(6BC) (lixo) (8DE)
x2 // x2 = 20 // x2 = 15	x1(7Ah)(9B7)(20) x2(20)(2DE) x3(6BC) (lixo) (8DE)
x3 // *x3 = 400 // x3 = &x2 == *x3 = 20 // *x3 = 15	x1(7Ah)(9B7)(20) x2(20)(2DE) x3(6BC) (400) (8DE)
	x1(7Ah)(9B7)(20) x2(20)(2DE) x3(2DE) (20) (8DE)
	x1(7Ah)(9B7)(20) x2(15)(2DE) x3(2DE) (15) (8DE)

- Faça o quadro de memória:

```
double M [3][3];
double *p = M[0];
for (int i = 0; i < pow(MAXTAM, 2); i++, p++){
    *p=0.0;
}
```

Memória
M[0][0] (7Ah) = 0.0
M[0][1] = 0.0
M[0][2] = 0.0
M[1][0] = 0.0
M[1][1] = 0.0
M[1][2] = 0.0
M[2][0] = 0.0
M[2][1] = 0.0
M[2][2] = 0.0
M[3][0] = 0.0
M[3][1] = 0.0
M[3][2] = 0.0
*p = 7Ah

- Mostre a saída na tela

<pre>double a; double *p, *q; a = 3.14; printf("%f\n", a); p = &amp;a; *p = 2.718; printf("%f\n", a); a = 5; printf("%f\n", *p);</pre>	<pre>p = NULL; p = (double*) malloc(sizeof(double)); *p = 20; q = p; printf("%f\n", *p); printf("%f\n", a); free(p); printf("%f\n", *q);</pre>
--	--

Tela:
3.14
2.718
5
20
5
lixo

- Mostre o quadro de memória

<pre>int a[10], *b; b = a; b[5] = 100; printf("\n%d -- %d", a[5], b[5]);  b = (int*) malloc(10*sizeof(int)); b[7] = 100; printf("\n%d -- %d", a[7], b[7]);  //O comando a = b gera um erro de compilação</pre>
--

Memória
a[0] = b[0] = lixo
a[1] = b[1] = lixo
a[2]= b[2] = lixo
a[3]= b[3] = lixo
a[4]= b[4] = lixo
a[5] = 100 (b[5] = 100) // = lixo
a[6] = b[5] = lixo
a[7] = b[6] = lixo
a[8] = b[7] = lixo // = 100
a[9] = b[8] = lixo

- Mostre o quadro de memória

```

int *x1;      int x2;      int *x3;
x1 = (int*) malloc(sizeof(int));
*x1 = 20;
x2 = *x1;
*x3 = x2 * *x1;
x3 = &x2;
x2 = 15;
x2 = 13 & 3;
x2 = 13 | 3;
x2 = 13 >> 1;
x2 = 13 << 1;

```

Memória
7Ab: *x1 = 20
x2 = 20 // x2 = 15 // x2 = 1 // x2 = 15 // x2 = 6 // x2 = 26
7Ad: *x3 = 400 // x3 -> x2 // *x3 = 15 //


- Represente graficamente o código Java abaixo

```
Elemento e1;
```

e1 -> null/?;


- Represente graficamente o código Java abaixo

```
Elemento e1 = new Elemento();
```

e1 -> 

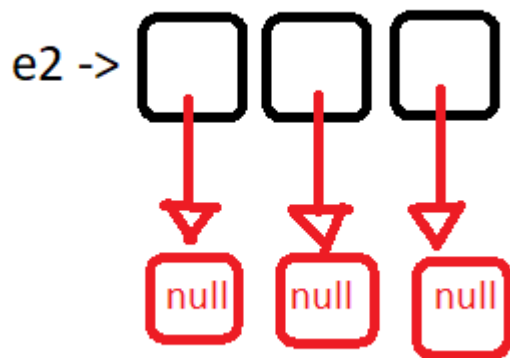
- Represente graficamente o código Java abaixo

```
Elemento[] e2 = new Elemento [3];
```

e2 -> 

- Represente graficamente o código Java abaixo

```
Elemento[] e2 = new Elemento [3];  
  
for (int i = 0; i < 3; i ++){  
    e2[i] = new Elemento();  
}
```



- Represente graficamente o código C abaixo

```
Elemento e1;
```

e1



- Represente graficamente o código C abaixo

```
Elemento* e2;
```

e2 -> null

- Represente graficamente o código C abaixo

```
Elemento* e2 = (Elemento*) malloc(sizeof(Elemento));
```

e2 ->

null

- Represente graficamente o código C abaixo

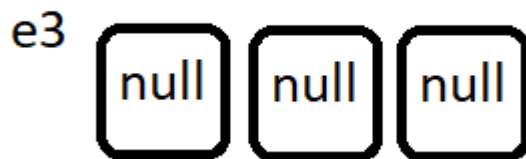
```
Elemento* e2 = (Elemento*) malloc(3*sizeof(Elemento));
```

e2 ->

null null null

- Represente graficamente o código C abaixo

```
Elemento e3[3];
```



- Represente graficamente o código C abaixo

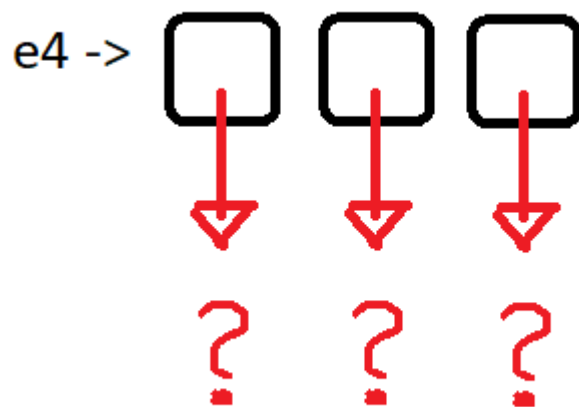
```
Elemento** e4;
```

e4 -> ?

- Represente graficamente o código C abaixo

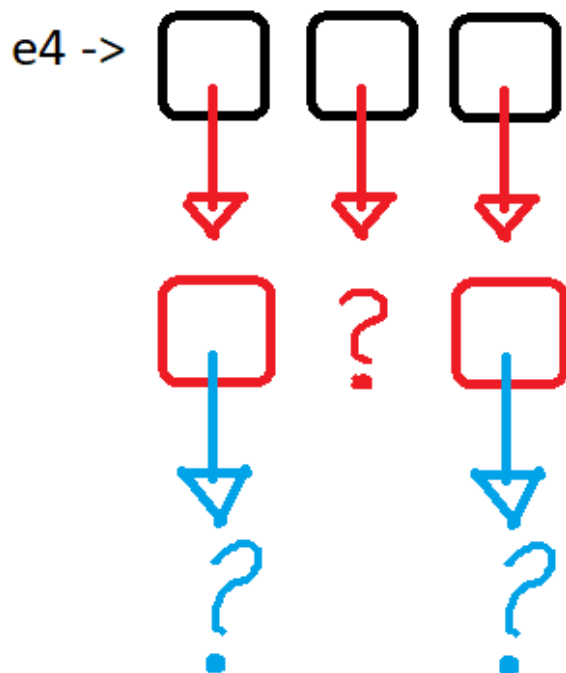
```
Elemento** e4 = (Elemento**) malloc(3*sizeof(Elemento*));
```






- Represente graficamente o código C abaixo

```
Elemento** e4 = (Elemento**) malloc(3*sizeof(Elemento*));  
e4[0] = (Elemento*) malloc(sizeof(Elemento*));  
e4[2] = (Elemento*) malloc(sizeof(Elemento*));
```




- Represente graficamente o código C++ abaixo

```
Elemento e1;
```

e1 


- Represente graficamente o código C++ abaixo

```
Elemento* e2;
```

e2 -> 


- Represente graficamente o código C++ abaixo

```
Elemento* e2 = new Elemento;
```

e2 -> 


- Represente graficamente o código C++ abaixo

```
Elemento* e2 = new Elemento[3];
```

e2 -> 

- Represente graficamente o código C++ abaixo

```
Elemento e3[3];
```

e3 

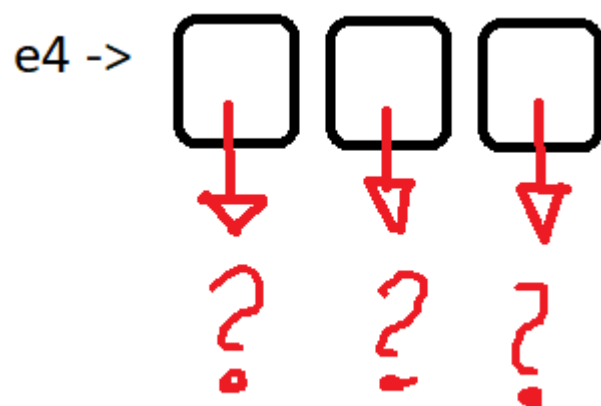
- Represente graficamente o código C++ abaixo

```
Elemento** e4;
```

e4 -> ?

- Represente graficamente o código C++ abaixo

```
Elemento** e4 = new Elemento*[3];
```



- Represente graficamente o código C++ abaixo

```
Elemento** e4 = new Elemento*[3];  
e4[0] = new Elemento;  
e4[2] = new Elemento;
```

