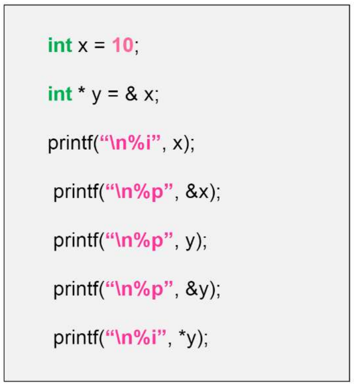
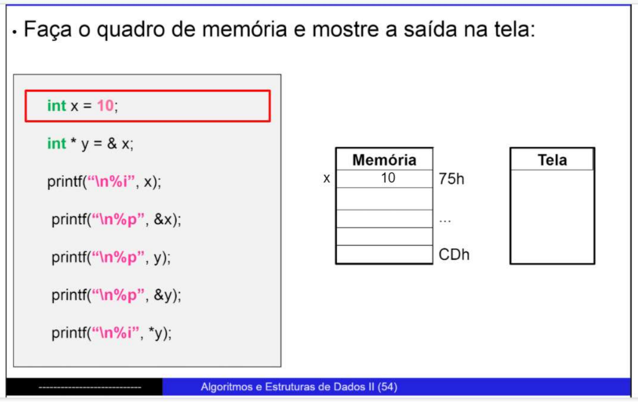
Exemplo



```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```



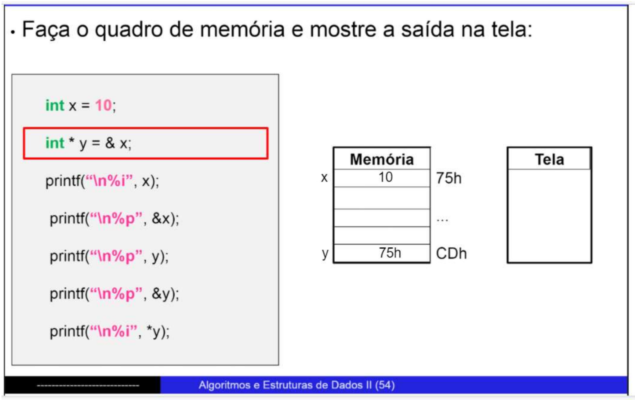
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
		CDh

Tela

------------------------- Algoritmos e Estruturas de Dados II (54)



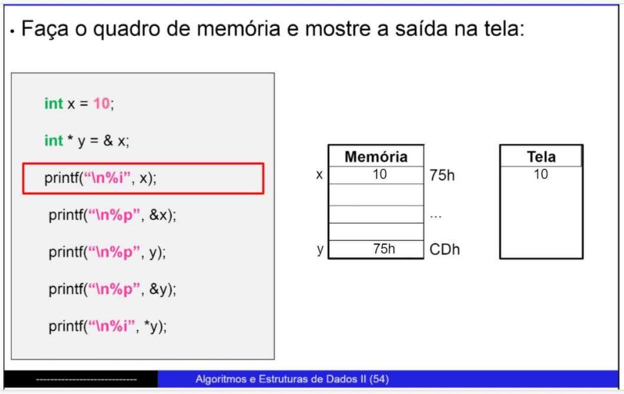
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
y	75h	CDh

Tela

------------------------- Algoritmos e Estruturas de Dados II (54)



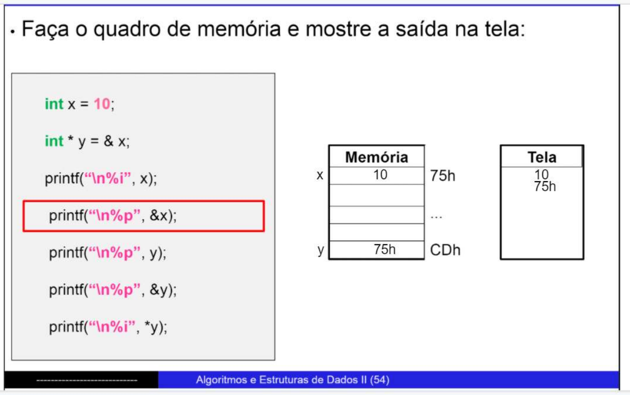
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
y	75h	CDh

Tela
10

------------------------- Algoritmos e Estruturas de Dados II (54)



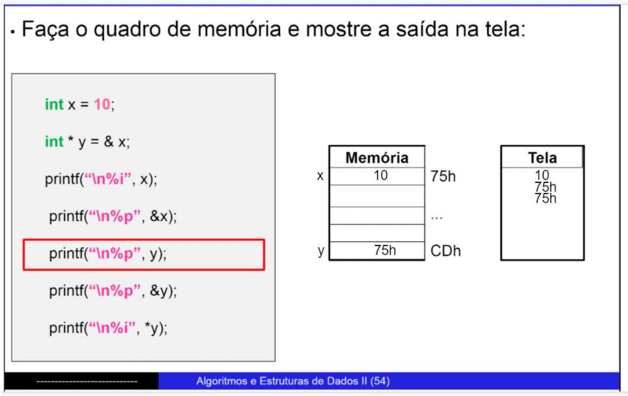
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
y	75h	CDh

Tela
10 75h

------------------------- Algoritmos e Estruturas de Dados II (54)



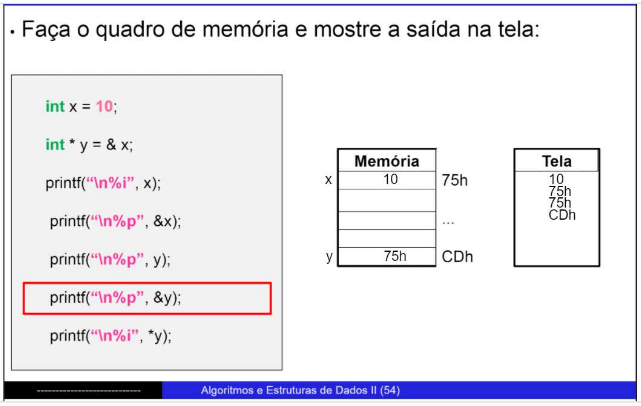
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
y	75h	CDh

Tela
10 75h 75h

------------------------- Algoritmos e Estruturas de Dados II (54)



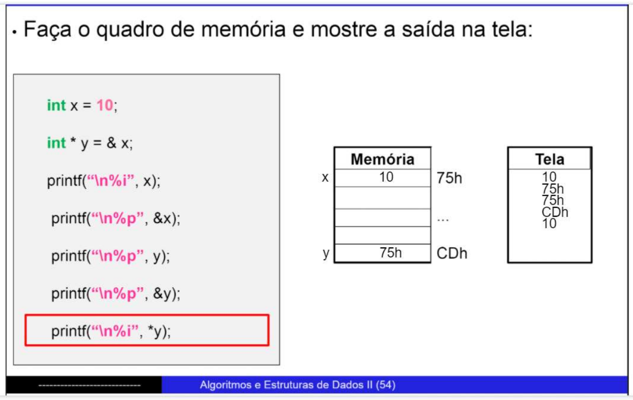
- Faça o quadro de memória e mostre a saída na tela:

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

	Memória	
x	10	75h
		...
y	75h	CDh

Tela
10 75h 75h CDh

------------------------- Algoritmos e Estruturas de Dados II (54)



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

```
int x = 10;

int * y = & x;

printf(“\n%i”, x);

 printf(“\n%p”, &x);

 printf(“\n%p”, y);

 printf(“\n%p”, &y);

 printf(“\n%i”, *y);
```

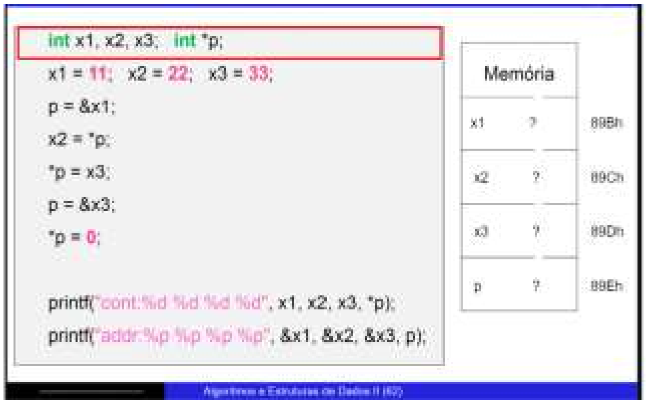
	Memória	
x	10	75h
		...
y	75h	CDh

Tela
10 75h 75h CDh 10

------------------------- Algoritmos e Estruturas de Dados II (54)

Exemplo

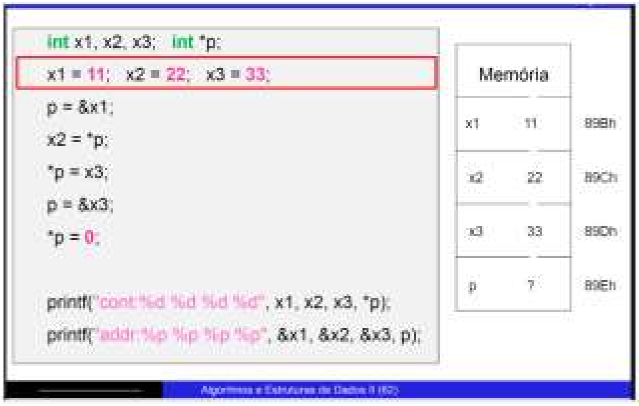
```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	?	89Bh
x2	?	89Ch
x3	?	89Dh
p	?	89Eh

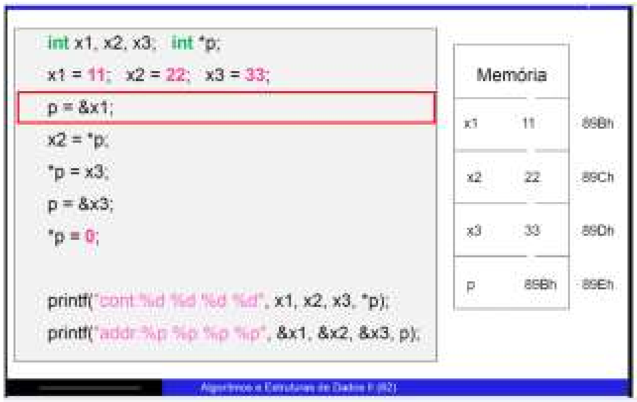
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	11	89Bh
x2	22	89Ch
x3	33	89Dh
p	?	89Eh

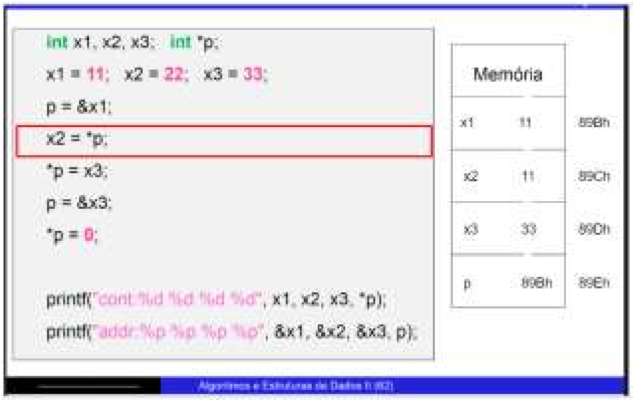
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	11	89Bh
x2	22	89Ch
x3	33	89Dh
p	89Bh	89Eh

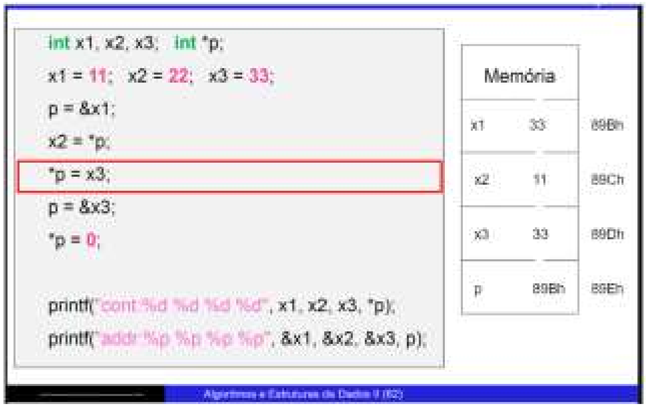
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	11	89Bh
x2	11	89Ch
x3	33	89Dh
p	89Bh	89Eh

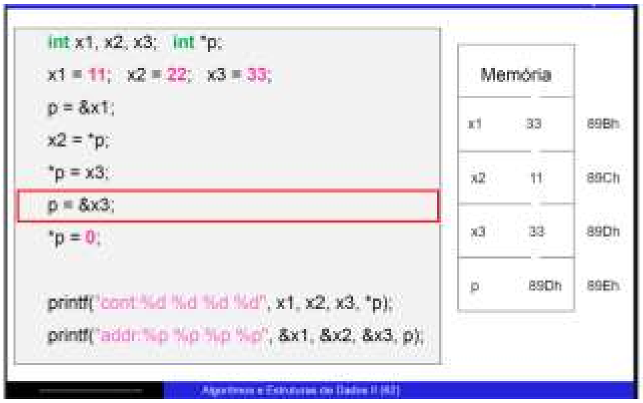
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	33	89Bh
x2	11	89Ch
x3	33	89Dh
p	89Bh	89Eh

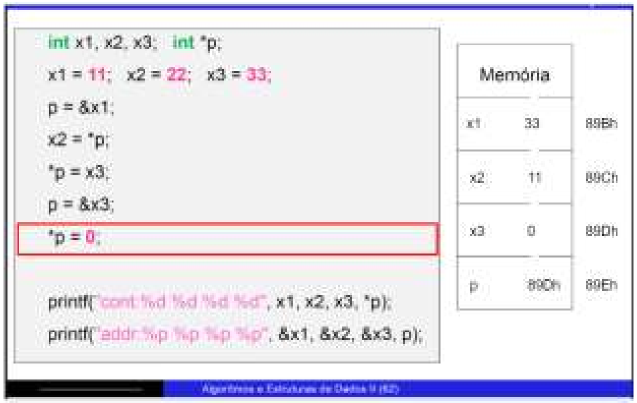
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	33	89Bh
x2	11	89Ch
x3	33	89Dh
p	89Dh	89Eh

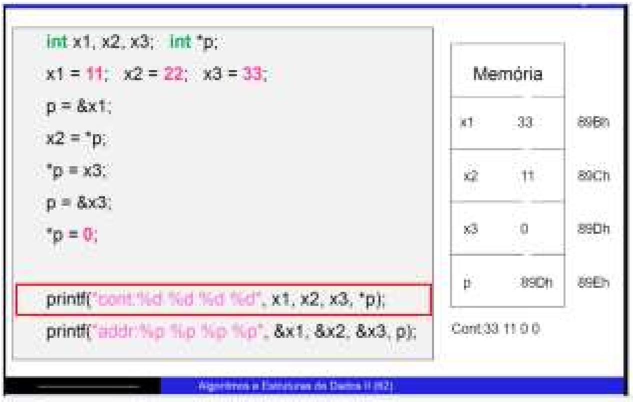
Algoritmos e Estruturas de Dados II (62)



```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	33	89Bh
x2	11	89Ch
x3	0	89Dh
p	89Dh	89Eh

Algoritmos e Estruturas de Dados II (62)

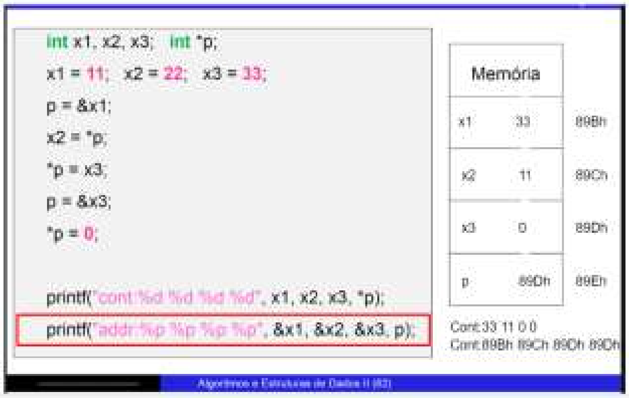


```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	33	89Bh
x2	11	89Ch
x3	0	89Dh
p	89Dh	89Eh

```
Cont:33 11 0 0
```

Algoritmos e Estruturas de Dados II (62)



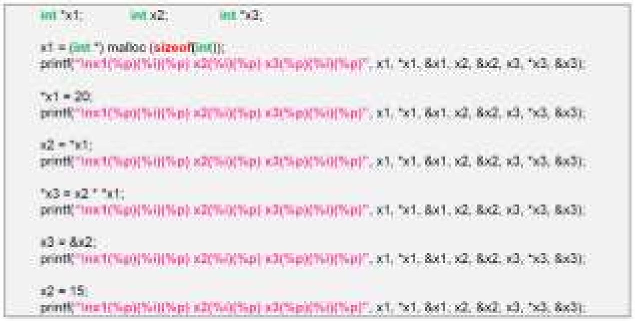
```
int x1, x2, x3;   int *p;
x1 = 11;   x2 = 22;   x3 = 33;
p = &x1;
x2 = *p;
*p = x3;
p = &x3;
*p = 0;

printf("cont:%d %d %d %d", x1, x2, x3, *p);
printf("addr:%p %p %p %p", &x1, &x2, &x3, p);
```

Memória		
x1	33	89Bh
x2	11	89Ch
x3	0	89Dh
p	89Dh	89Eh

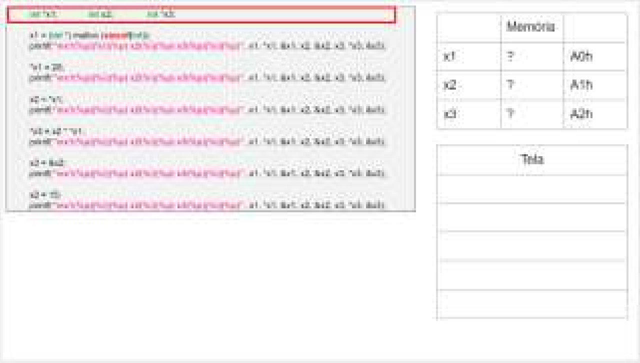
```
Cont:33 11 0 0
Cont:89Bh 89Ch 89Dh 89Dh
```

Algoritmos e Estruturas de Dados II (62)

Exemplo



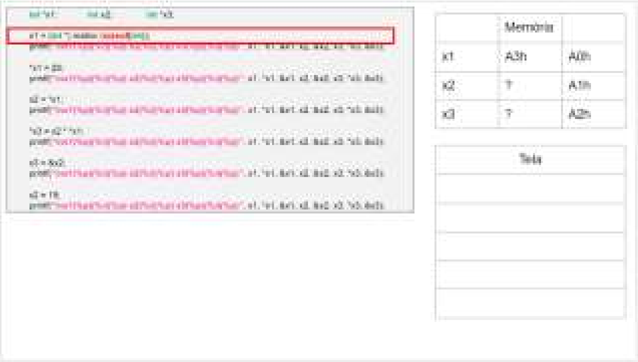
```
int *x1;          int x2;          int *x3;

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);
```



```
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);
```

	Memoria	
x1	?	A0h
x2	?	A1h
x3	?	A2h

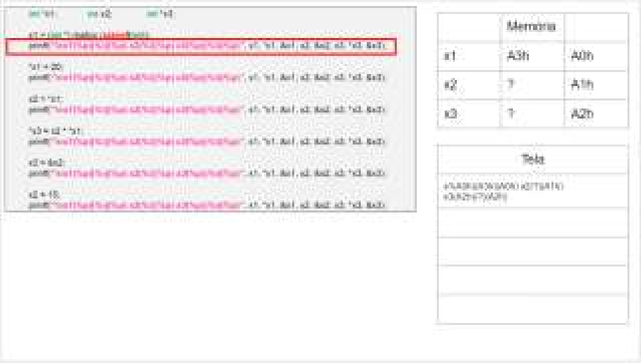
Tela



```
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);
```

	Memoria	
x1	A3h	A0h
x2	?	A1h
x3	?	A2h

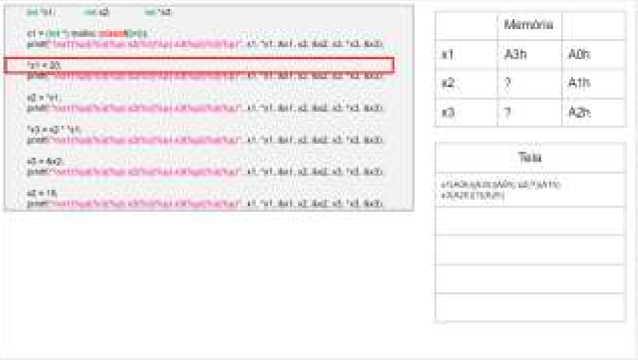
Tela



```
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);
```

	Memoria	
x1	A3h	A0h
x2	?	A1h
x3	?	A2h

Tela
[illegible]

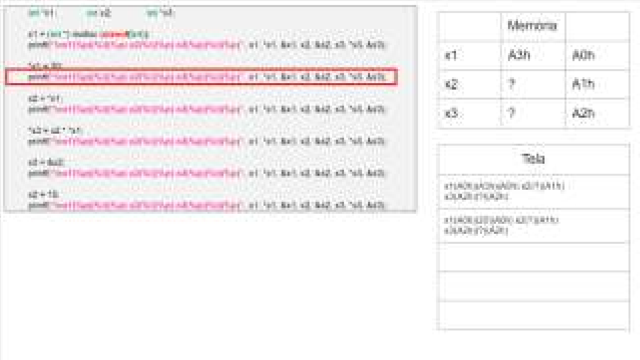


```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

	Memória	
x1	A3h	A0h
x2	?	A1h
x3	?	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)



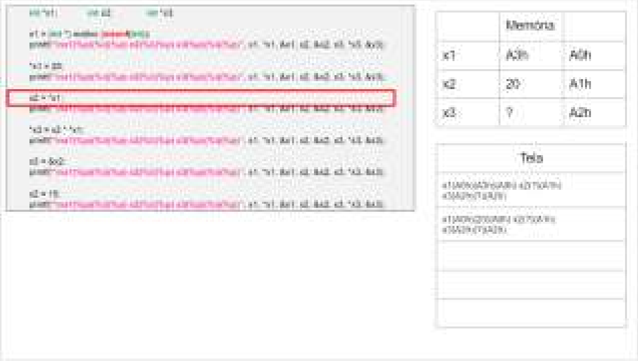
```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

	Memória	
x1	A3h	A0h
x2	?	A1h
x3	?	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)



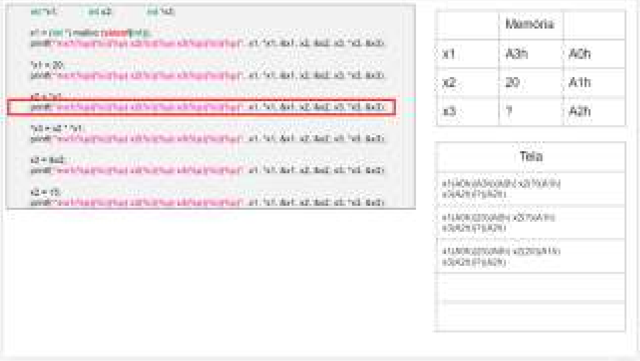
```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

	Memória	
x1	A3h	A0h
x2	20	A1h
x3	?	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)



```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

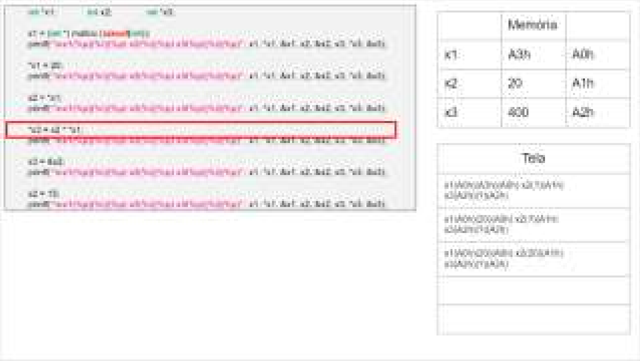
	Memória	
x1	A3h	A0h
x2	20	A1h
x3	?	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(20)(A1h)
x3(A2h)(?)(A2h)



```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

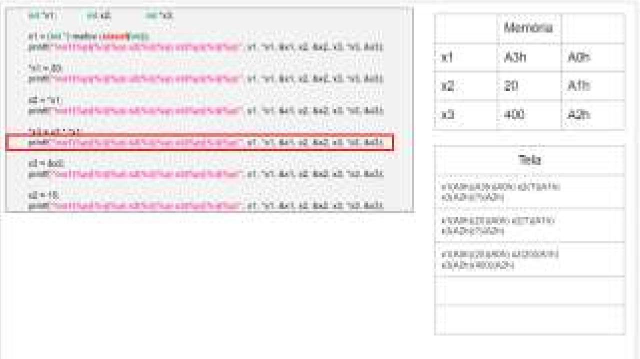
	Memória	
x1	A3h	A0h
x2	20	A1h
x3	400	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(20)(A1h)
x3(A2h)(?)(A2h)



```
int *x1;        int x2;        int *x3;

x1 = (int *) malloc (sizeof(int));
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x1 = 20;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

*x3 = x2 * *x1;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x3 = &x2;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);

x2 = 15;
printf("[illegible]", x1, *x1, &x1, x2, &x2, x3, *x3, &x3);
```

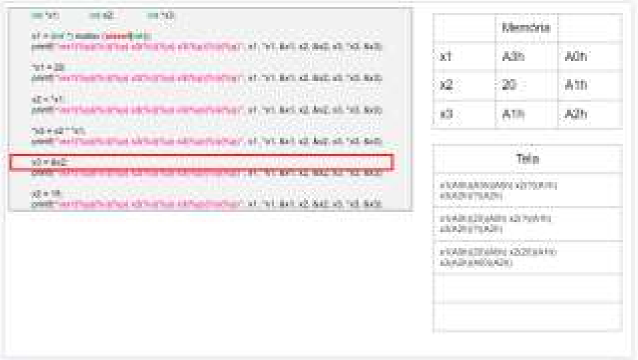
	Memória	
x1	A3h	A0h
x2	20	A1h
x3	400	A2h

Tela

x1(A0h)(A3h)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

x1(A0h)(20)(A0h) x2(?)(A1h)
x3(A2h)(?)(A2h)

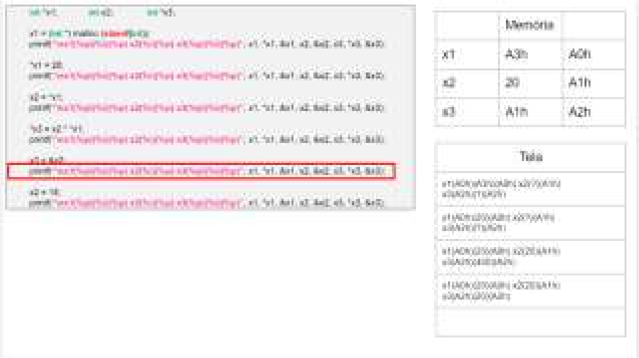
x1(A0h)(20)(A0h) x2(20)(A1h)
x3(A2h)(400)(A2h)



```
int *x1;        int x2;        int *x3;

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

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

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

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

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

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

	Memória	
x1	A3h	A0h
x2	20	A1h
x3	A1h	A2h

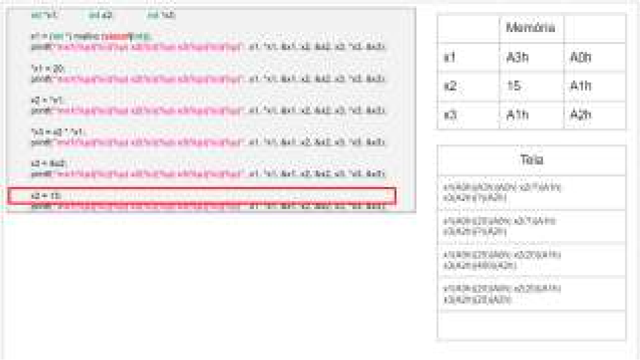
Tela
x1(A0h)(A3h)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A2h)(400)(A2h)



```
int *x1;        int x2;        int *x3;

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

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

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

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

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

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

	Memória	
x1	A3h	A0h
x2	20	A1h
x3	A1h	A2h

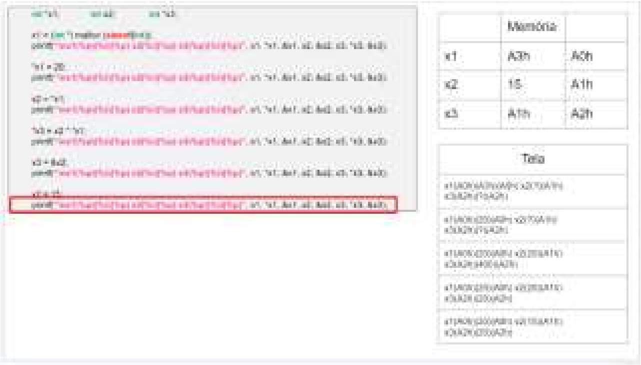
Tela
x1(A0h)(A3h)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A2h)(400)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A1h)(20)(A2h)



```
int *x1;        int x2;        int *x3;

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

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

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

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

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

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

	Memória	
x1	A3h	A0h
x2	15	A1h
x3	A1h	A2h

Tela
x1(A0h)(A3h)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A2h)(400)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A1h)(20)(A2h)

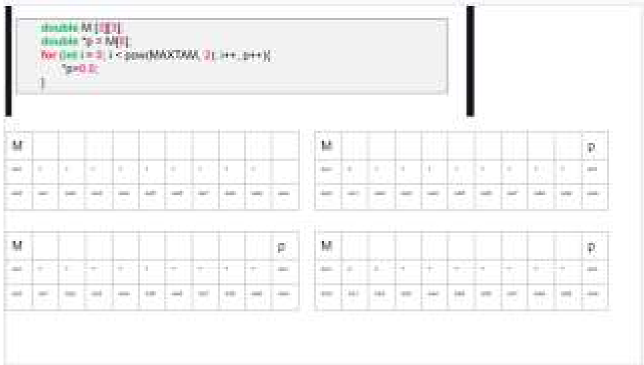


```
int *x1;        int x2;        int *x3;

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

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

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

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

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

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

	Memória	
x1	A3h	A0h
x2	15	A1h
x3	A1h	A2h

Tela
x1(A0h)(A3h)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(7)(A1h) x3(A2h)(7)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A2h)(400)(A2h)
x1(A0h)(20)(A0h) x2(20)(A1h) x3(A1h)(20)(A2h)
x1(A0h)(20)(A0h) x2(15)(A1h) x3(A1h)(15)(A2h)

Exercício



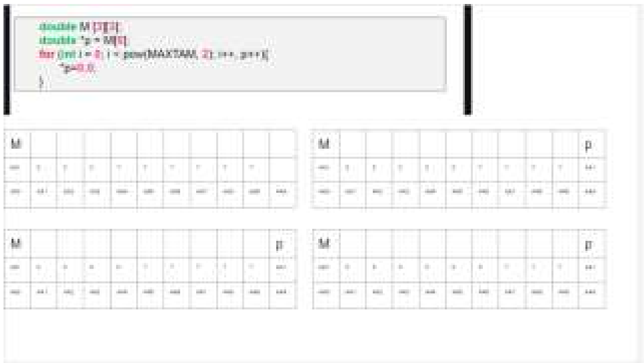
```
double M [3][3];
double *p = M[0];
for (int i = 0; i < pow(MAXTAM, 2); i++, p++){
    *p=0.0;
}
```

M										
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



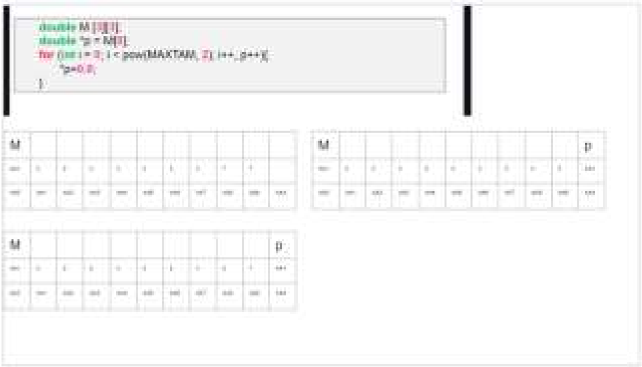
```
double M [3][3];
double *p = M[0];
for (int i = 0; i < pow(MAXTAM, 2); i++, p++){
    *p=0.0;
}
```

M										
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



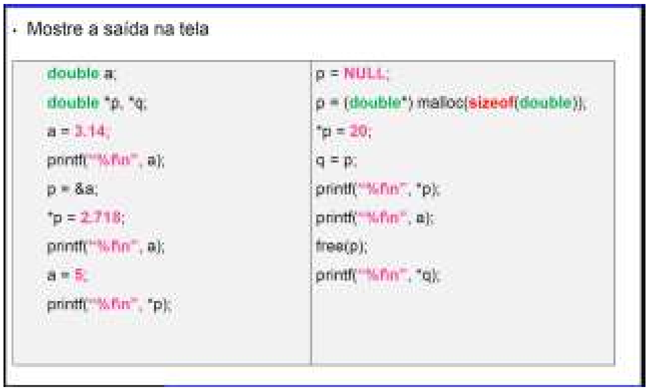
```
double M [3][3];
double *p = M[0];
for (int i = 0; i < pow(MAXTAM, 2); i++, p++){
    *p=0.0;
}
```

M										
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

M										p
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Exercício



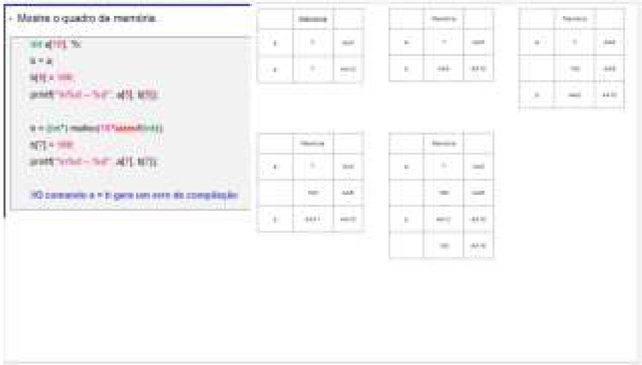
- Mostre a saída na tela

```
double a;
double *p, *q;
a = 3.14;
printf("%f\n", a);
p = &a;
*p = 2.718;
printf("%f\n", a);
a = 5;
printf("%f\n", *p);
```

```
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);
```

```
3.14
2.718
5
20
5
NULL
```

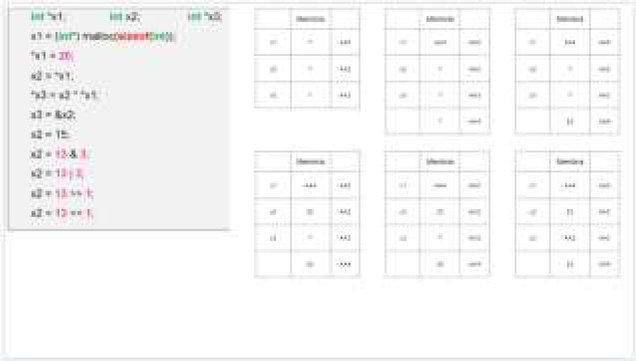
Exercício



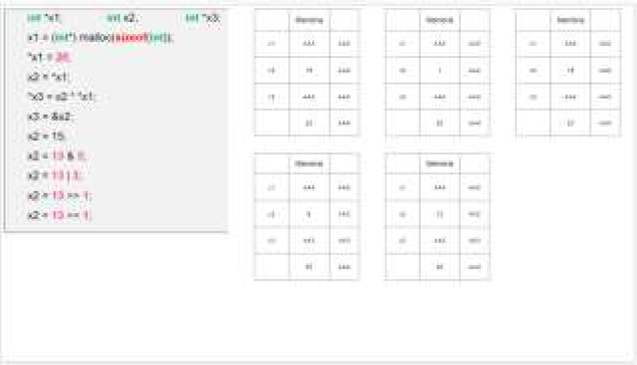
- Mostre o quadro de memória

```
int a[10], *b;
b = a;
b[5] = 100;
printf("%d -- %d", a[5], b[5]);

b = (int*) malloc(10*sizeof(int));
b[7] = 100;
printf("%d -- %d", a[7], b[7]);

//O comando a = b gera um erro de compilação
```

Exercício



```
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;
```



```
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;
```

Exercício

- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

```
[illegible]
```

Representação gráfica

- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

```
[illegible]
```

Representação gráfica

- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

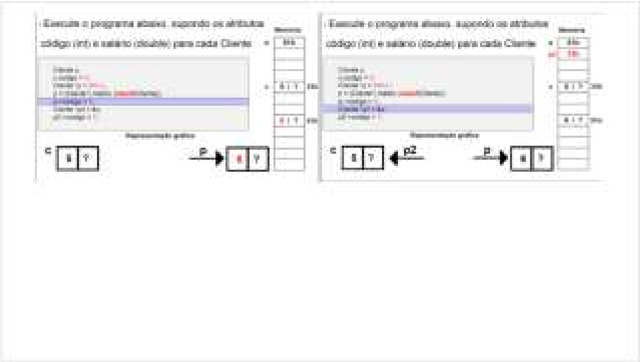
```
[illegible]
```

Representação gráfica

- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

```
[illegible]
```

Representação gráfica



- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

```
[illegible]
```

Representação gráfica

- Execute o programa abaixo, supondo os atributos código (int) e salário (double) para cada Cliente

```
[illegible]
```

Representação gráfica