



Matrizes

- Também conhecidos como arrays, vetores
- Conjunto de informações do mesmo tipo
- Variável com mais de uma posição (slot)
- Índice numérico sempre no zero
- Matrizes unidimensionais (uma dimensão)
- Sintaxe: `tipo nome[tamanho];`

```
int numeros[5];
numeros[0] = 12;
numeros[1] = 54;
numeros[2] = 6;
```

12	54	6		
[0]	[1]	[2]	[3]	[4]

```
int numeros[7] = {33, 34, 35, 36, 37, 44};
```

33	34	35	36	37	44	
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Tamanhos de matriz

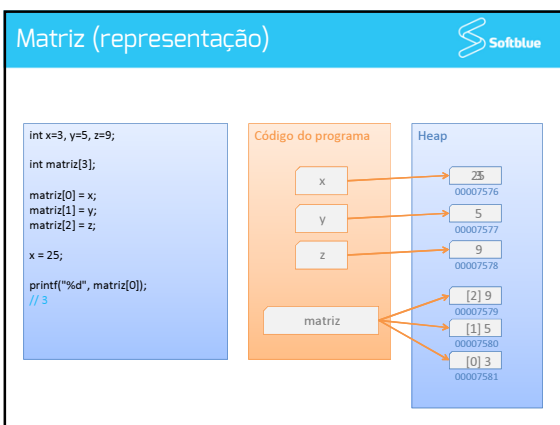
- Instrução **sizeof**
 - Obtém o tamanho em bytes
 - Sabendo o tamanho total, e o tamanho do tipo de dado, pode-se calcular sua quantidade de posições

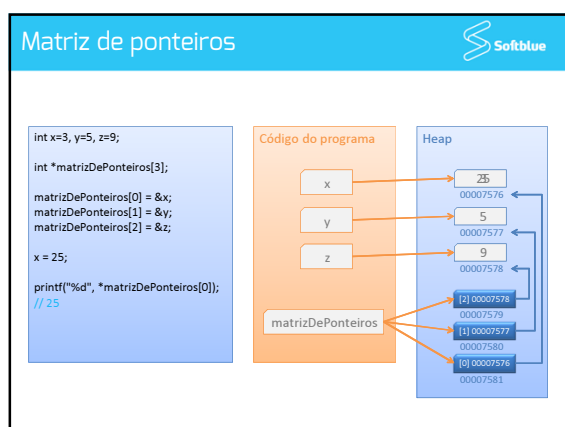
```
int numeros[7] = {33, 34, 35, 36, 37, 44};

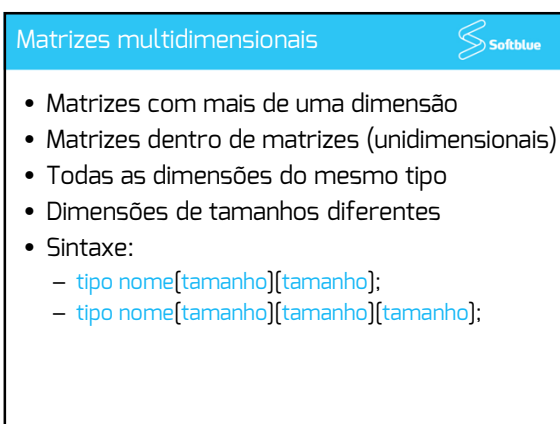
int tamanhoBytes = sizeof(numeros);
printf("Minha matriz tem %d bytes", tamanhoBytes);
// Minha matriz tem 28 bytes

int tamanhoPosicoes = sizeof(numeros) / sizeof(int);
printf("Minha matriz tem %d posicoes", tamanhoPosicoes);
// Minha matriz tem 7 posicoes
```


33	34	35	36	37	44	
[0]	[1]	[2]	[3]	[4]	[5]	[6]







Matrizes multidimensionais



- Exemplo

```
int dobro3[3][2];
dobro3[0][0] = 2;
dobro3[0][1] = 4;


dobro3[1][0] = 10;
dobro3[1][1] = 20;

dobro3[2][0] = 50;
dobro3[2][1] = 100;

printf("O dobro de %d: %d", dobro3[2][0], dobro3[2][1]);
// O dobro de 50: 100
```

2	4	10	20	50	100
[0]	[1]	[0]	[1]	[0]	[1]
[0]		[1]		[2]	

Alocação dinâmica de memória



- Instrução malloc em matrizes

```
int main()
{
    int *matriz, x;

    matriz = malloc(5 * sizeof(int));

    matriz[0] = 10;
    matriz[1] = 15;
    matriz[2] = 20;
    matriz[3] = 25;
    matriz[4] = 30;

    for(x=0; x<5; x++)
    {
        printf("%d ", matriz[x]);
    }

    // 10 15 20 25 30

    free(matriz);

    return 0;
}
```



Softblue
