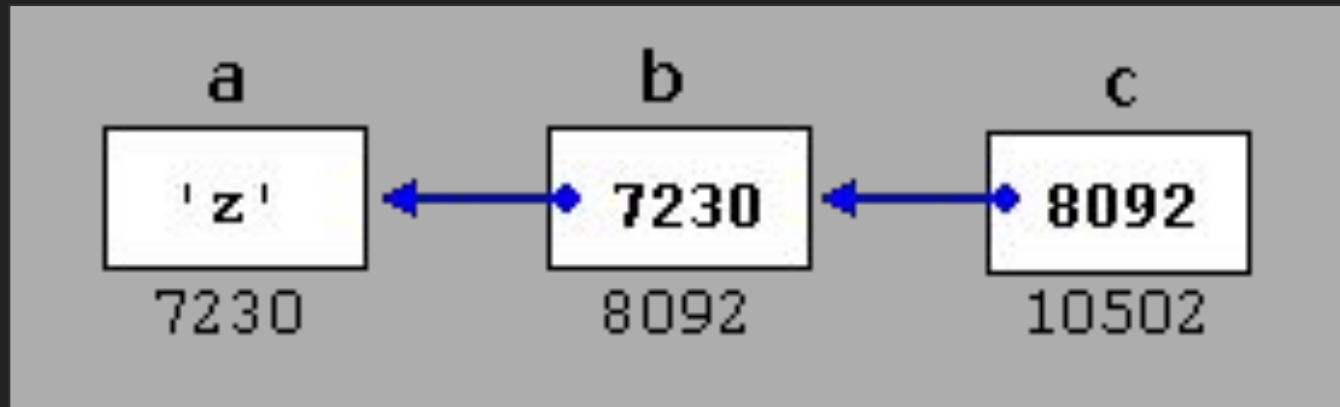


Ponteiros e Alocação Dinâmica Multidimensionais

Prof.: Leonardo Tórtoro Pereira

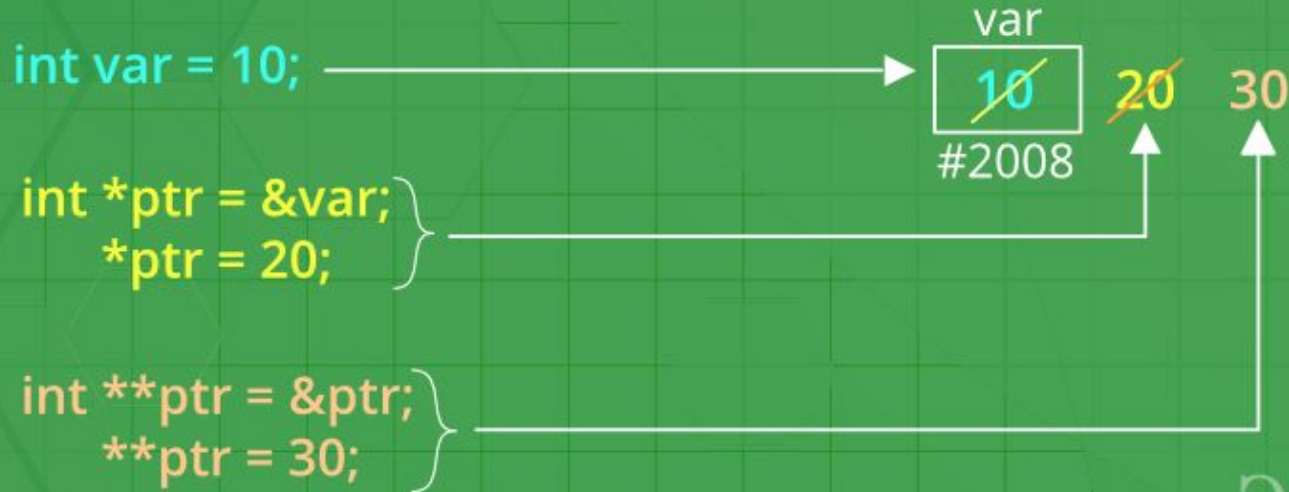
leonardop@usp.br

Ponteiros?



Fonte: <http://www.cplusplus.com/doc/tutorial/pointers/>

How pointer works in C

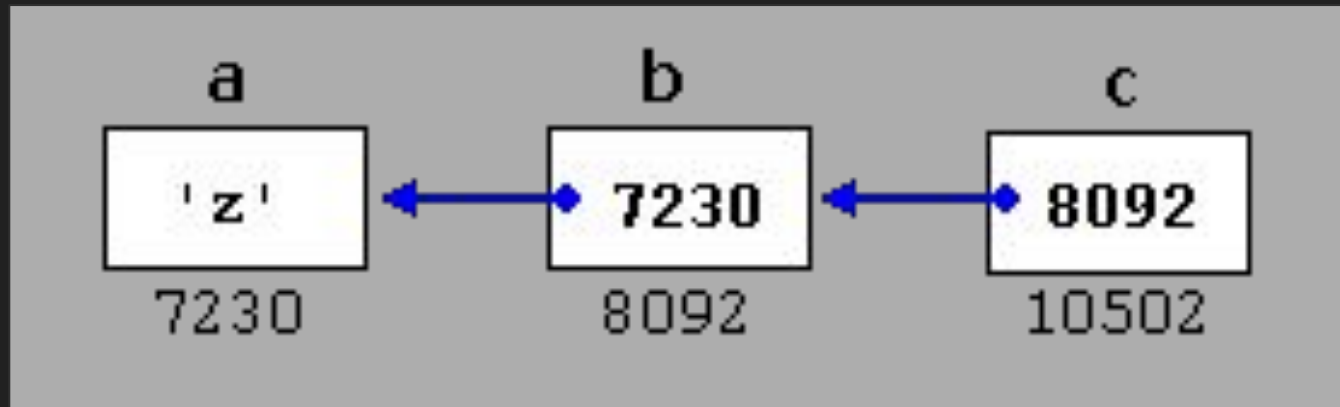


Ponteiros de ponteiros

Ponteiros de ponteiros

- Ponteiros podem apontar para outros ponteiros
 - ◆ E assim sucessivamente...
 - ◆ Cada novo nível requer um * a mais na declaração

```
char a;  
char * b;  
char ** c;  
a = 'z';  
b = &a;  
c = &b;
```



Fonte: <http://www.cplusplus.com/doc/tutorial/pointers/>

Ponteiros de ponteiros

- No exemplo anterior temos os seguintes tipos e valores
 - ◆ *c* é um *char*** com valor 8092
 - ◆ **c* é um *char** com valor 7230
 - ◆ ***c* é um *char* com valor 'z'
- No geral, eles são equivalentes a vetores multidimensionais
- Considere a declaração seguinte:
 - ◆ `int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };`

Ponteiros de ponteiros

Anotação de ponteiro	Anotação de vetor	Valor
<code>**nums</code>	<code>nums[0][0]</code>	16
<code>*(*nums+1)</code>	<code>nums[0][1]</code>	18
<code>*(*nums+2)</code>	<code>nums[0][2]</code>	20
<code>*(*(nums+1)</code>	<code>nums[1][0]</code>	25
<code>*(*(nums+1)+1)</code>	<code>nums[1][1]</code>	26
<code>*(*(nums+1)+2)</code>	<code>nums[1][2]</code>	27

Ponteiros de ponteiros

```
int main () {  
    int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };  
    int **numsp;  
    numsp = (int**) malloc(sizeof(int*)*2);  
    for(int i = 0; i < 2; ++i)  
        numsp[i] = (int*)malloc(sizeof(int)*3);  
    printf("\nVetor:\n");  
    for(int i = 0; i < 2; ++i) {  
        for(int j = 0; j < 3; ++j) {  
            printf("%d - ", nums[i][j]);  
            *(*(numsp+i)+j) = nums[i][j];  
        }  
        printf("\n");  
    }  
}
```

Ponteiros de ponteiros

```
printf("\nPonteiro:\n");
for(int i = 0; i < 2; ++i) {
    for(int j = 0; j < 3; ++j) {
        printf("%d - ", (*(numsp+i)+j));
    }
    printf("\n");
}

for(int i = 0; i < 2; ++i)
    free(numsp[i]);
free(numsp);
}
```

Ponteiros de ponteiros

- Podemos fazer quantas “dimensões” quisermos de ponteiros
- Inclusive para ler dinamicamente strings
- Vamos ao exemplo!

Passando arrays multidimensionais para funções

Array Multidimensional em Funções

- É um pouco complicado passar arrays multidimensionais para funções
 - ◆ É preciso especificar o tamanho de qualquer dimensão maior que 1

Array Multidimensional em Funções

- `print(int arr[M][N])`
 - ◆ Sendo M e N globais
- `print(int arr[][N], int m)`
 - ◆ Sendo N global
- A partir do C99, é possível passar como uma variável
 - ◆ `print(int m, int n, int arr[][n])`

Array Multidimensional em Funções

→ Uma alternativa é passá-lo como um ponteiro

```
void print(int *arr, int m, int n) {  
    int i, j;  
    for (i = 0; i < m; i++)  
        for (j = 0; j < n; j++)  
            printf("%d ", *((arr+i*n) + j));  
}  
  
int main() {  
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};  
    int m = 3, n = 3;  
    // We can also use "print(&arr[0][0], m, n);"  
    print((int *)arr, m, n);  
    return 0;  
}
```


Referências

1. <http://www.cplusplus.com/doc/tutorial/pointers/>
2. <https://www.geeksforgeeks.org/pointer-vs-array-in-c/>
3. <https://www.geeksforgeeks.org/pass-2d-array-parameter-c/>