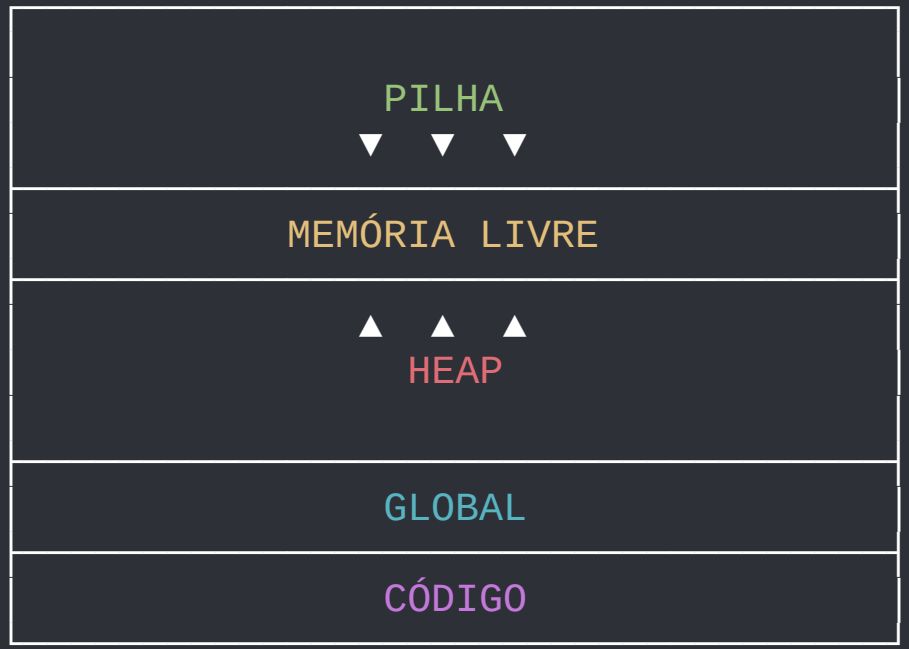
Ponteiros

Ponteiros

Ponteiros são *variáveis* capaz de armazenar endereços de memória. Usamos ponteiros quando precisamos manipular a memória diretamente. A memória RAM é divida em quatro partes:

- Área do código, onde o programa compilado reside;
- Área global, onde residem as variáveis globais;
- Área da pilha, onde os parâmetros de funções e variáveis locais residem;
- Área da heap, onde as variáveis dinâmicas residem.

Ponteiros

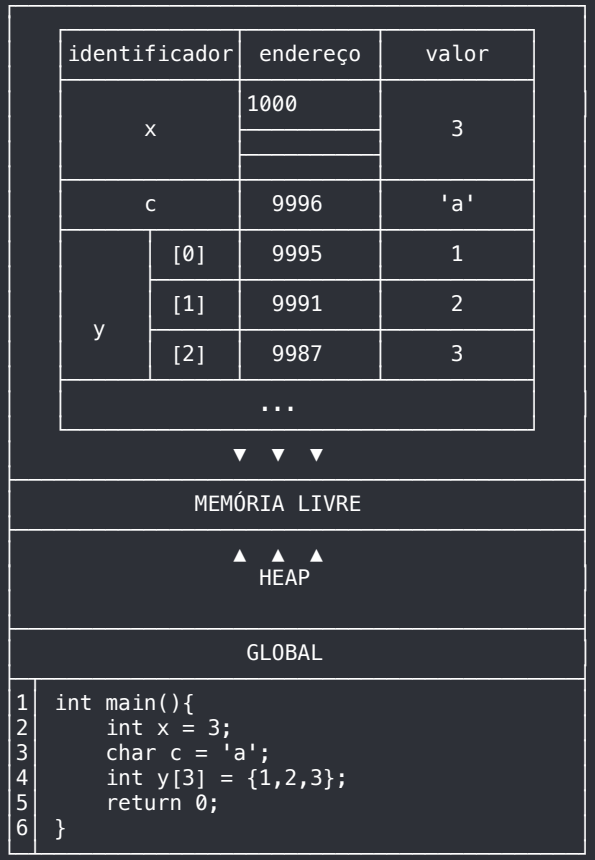


- a função `main()` e todas as funções ficam na pilha;
- O espaço necessário para cada variável depende do seu tipo.

Vamos supôr declaração de variáveis do código a abaixo conforme tabela:

```c
#include <stdio.h>
int main(){
    int x = 3;
    char c = 'a';
    int y[3] = {1,2,3};
    return 0;
}
```

Ponteiros



Referência:

```
myvar = 25;
foo = &myvar;
bar = myvar;
```

Desreferencia:

```
baz = *foo;
```

Uma forma interessante para entendermos ponteiros é correlacionando ao conceito de link simbólico em sistemas tipo Unix.

✔ Vamos para exemplos prático

Ilustrando a operação do exemplos, seria mais ou menos para fins didáticos:

identificador		endereço	valor
x		1000	3
c		9996	'a'
y	[0]	9995	1
	[1]	9991	2
	[2]	9987	3
p		9987	1000

1. O tipo dos ponteiros precisam ser do mesmo tipo da variável que ele está apontando;
2. Ponteiros de arrays, precisam apontar para um elemento do array;
3. Se passássemos o conteúdo de letra para `pletra` , imprimiria SOMENTE o primeiro elemento, imprimindo COM o asterisco na frente do ponteiro `*pletra`;
4. Se passássemos o conteúdo de letra para `pletra` , imprimiria TODOS elementos, imprimindo SEM o asterisco na frente do ponteiro `pletra`;
5. Se quiséssemos imprimir o terceiro elemento de índice 2 , mas igualando como no exemplo anterior

Sempre que puder usar ponteiro, use-o!

Como funciona os Ponteiros

É importantíssimo entender ponteiros, pois ele é uma peça chave para o desempenho dos seu programas C.

```
char * string = “Meu conteúdo”;
```

```
char * string = “Meu conteúdo”;
char * string2 = string;
```

```
string2 = “Meu conteúdo”;
```

```
char * string = “Meu conteúdo”;
```

```
char * string = “Meu conteúdo”;
char ** string2 = &string;
```

```
string2 = 0x74521gf56;
```