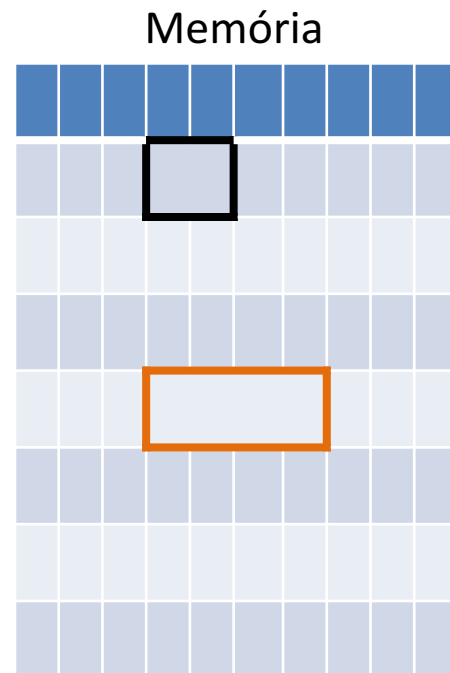


Estrutura de Dados e Algoritmos

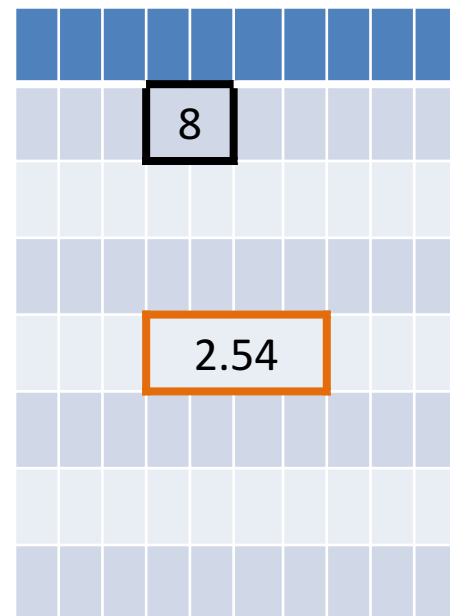
Ponteiros

- Quando se declara uma variável, define-se seus seguintes atributos:
 - Nome, Tipo (explícito)
 - **Endereço (implícito)**
- `int n;`
- `float m;`



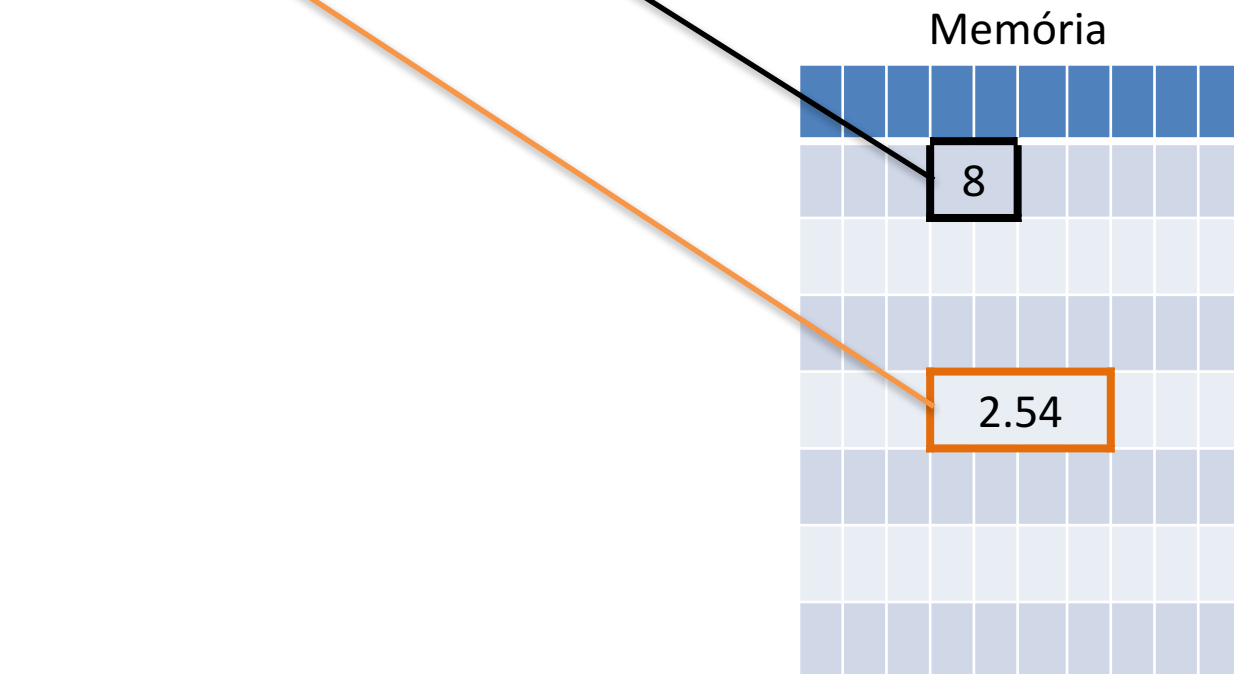
- $N = 8$;
- $m = 2.54$;

Memória

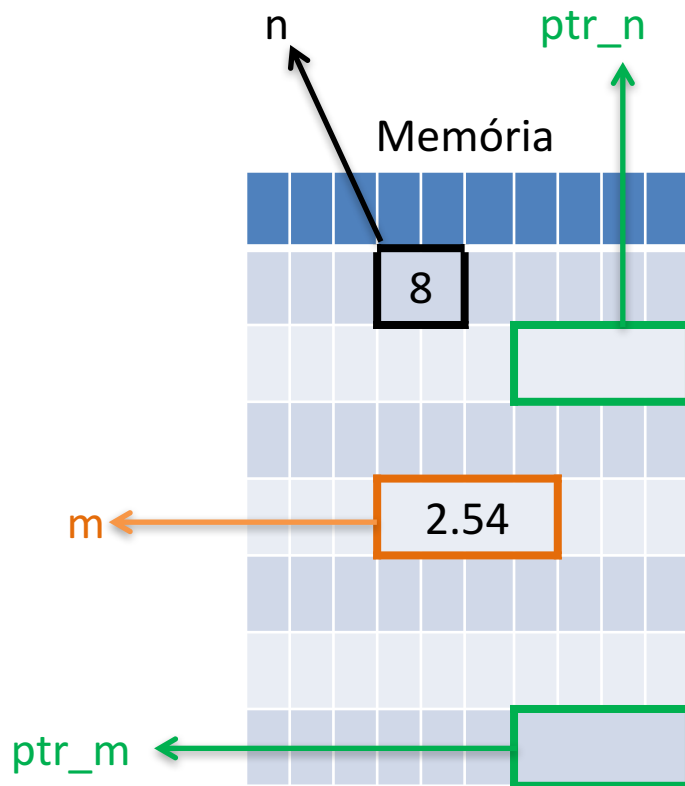


Endereço de Memória

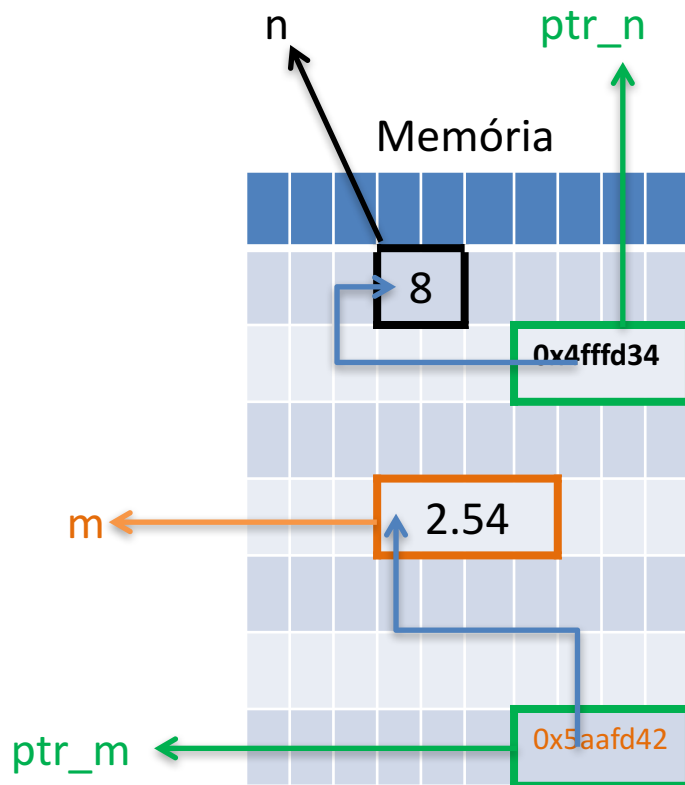
- $\&N \rightarrow 0x4fffd34$
- $\&m \rightarrow 0x5aafd42$



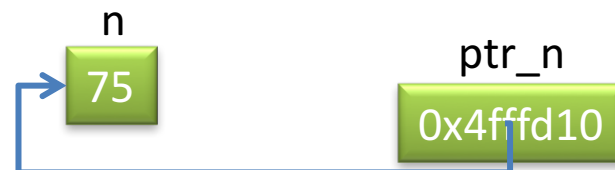
- Ponteiro: uma variável que armazena um endereço de memória;
- `int *ptr_n;`
- `float *ptr_m;`



- `ptr_n = &n;`
- `ptr_m = &m;`



- Exemplos:



```
void main()
{
    int n = 75;
    int *ptr_n;
    ptr_n = &n;
    printf("O valor da variável n é %d e seu endereço é %x\n", n, &n);
    printf("O endereço apontado pelo ponteiro ptr_n é %x e o valor do conteúdo deste endereço é %d\n", ptr_n, *ptr_n);
}
```

O valor da variável n é 75 e seu endereço é 0x4fffd10

O endereço apontado pelo ponteiro ptr_n é 0x4fffd10 e o valor do conteúdo deste endereço é 75

- Exemplos:

```
void main()
{
    int n = 75;
    int *ptr_n;
    ptr_n = &n;
    n = 31;
    printf("O valor da variável n é %d e seu endereço é %x\n", n, &n);
    printf("O endereço apontado pelo ponteiro ptr_n é %x e o valor do conteúdo deste endereço é %d\n", ptr_n, *ptr_n);
}
```

O valor da variável n é ?? e seu endereço é 0x4fffd10

O endereço apontado pelo ponteiro ptr_n é 0x4fffd10 e o valor do conteúdo deste endereço é ??

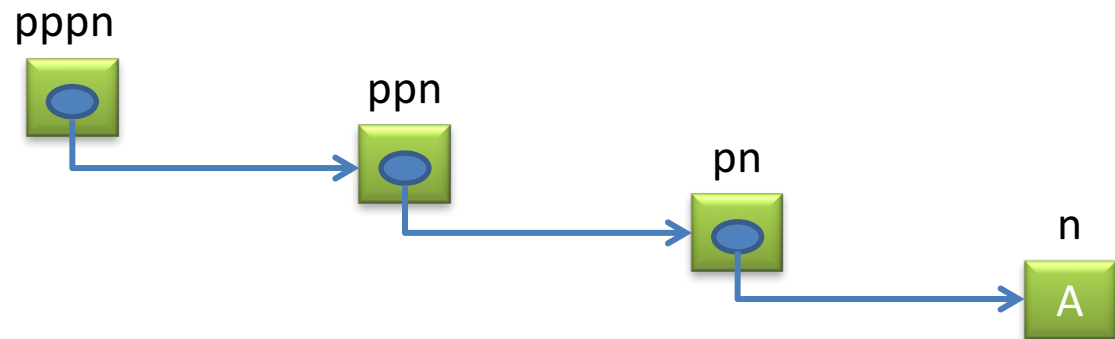
- & - Obtém o endereço de uma variável;
`int n;`
`&n → 0x4fffd10`
- * - Declara uma variável como ponteiro;
`int *n;`
- * - Obtém o conteúdo de uma variável ponteiro
`int n, *ptr_n;`
`n = 22;`
`ptr_n = &n;`
`*ptr_n → 22;`

```
char n= 'A', *pn, **ppn, ***pppn;
```

```
pn = &n;
```

```
ppn = &pn;
```

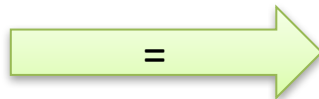
```
pppn = &ppn;
```



*pn ?

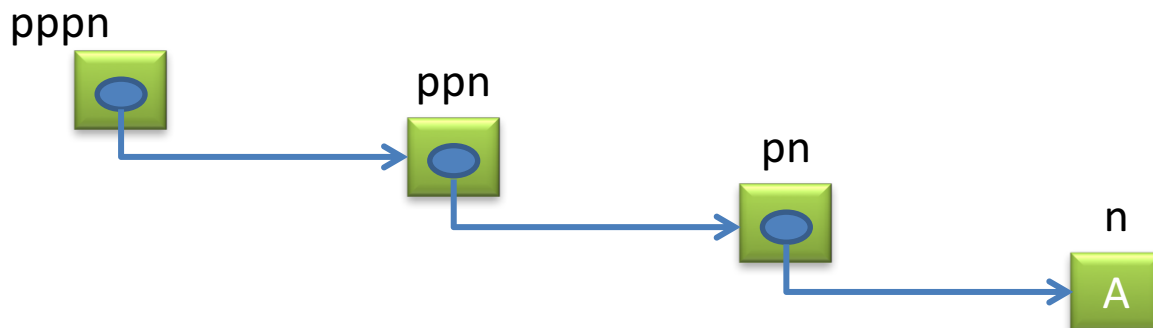
**ppn ?

***pppn ?

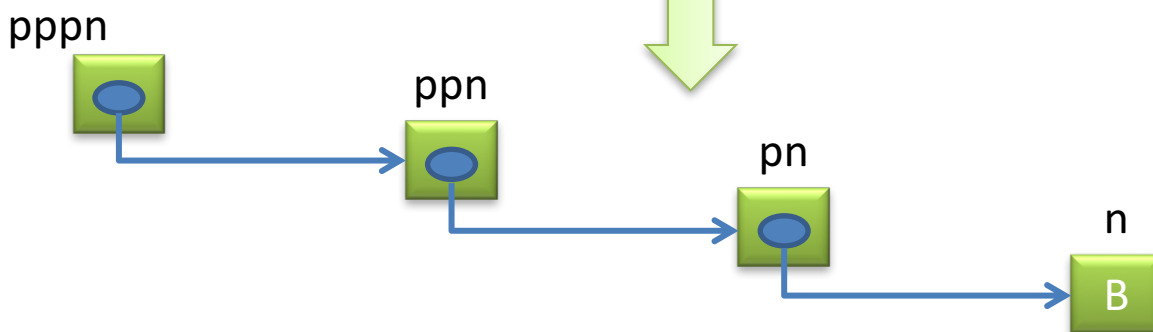


'A'

Ponteiros de Ponteiros



***`pppn = 'B';`



1. Faça um programa que leia 2 inteiros, armazene-os em nas variáveis *numa* e *numb* e imprima: 1. os valores lidos e os endereços de *numa* e *numb*.
2. Refaça o programa declarando *numa* e *numb* como ponteiros.

- Matrizes são coleções de objetos de um mesmo tipo, referenciadas por um nome comum;
- A sintaxe para a declaração de matrizes é: tipo do dado nome da matriz[N1][N2] . . . [Nm]; onde N_i é o número de elementos na i -ésima dimensão.
- Uma matriz com apenas uma dimensão é denominada vetor:
- Exemplos:
 - `int lista[100], tabela[12][15];`
 - `char nome[30], nomes[50][30];`
 - `float alturas[25];`

- Os elementos de uma matriz são acessados através de índices, que indicam as coordenadas do elemento na matriz.
- Em C/C++, diferentemente da matemática, os índices das matrizes começam em zero, não em 1 (um);

```
void main()
{
    int quadrados[5];
    for(int ind=0; ind<5; ind++)
        quadrados[ind] = ind*ind;
}
```

```
quadrados[0] = 0
quadrados[1] = 1
quadrados[2] = 4
quadrados[3] = 9
quadrados[4] = 16
```

- É possível inicializar uma matriz durante sua declaração, deixando a dimensão em aberto (sem preencher) e listando os elementos entre chaves e separados por vírgulas.

```
void main()  
{  
    int quadrados[] = {0, 1, 4, 9, 16};  
}
```

```
quadrados[0] = 0  
quadrados[1] = 1  
quadrados[2] = 4  
quadrados[3] = 9  
quadrados[4] = 16
```

- Em C uma string é um tipo especial de vetor:

```
void main()
{
    char nome[] = "Maria";
}
```

Elemento	Conteúdo
nome[0]	M
nome[1]	a
nome[2]	r
nome[3]	i
nome[4]	a

- O nome de um vetor é um ponteiro.

```
void main()
{
    int lista[5] = {10, 20, 30, 40, 50};
}
```

Índice	Memória	Endereço/Ponteiro
0	10	*lista
1	20	*(lista+1)
2	30	*(lista+2)
3	40	*(lista+3)
4	50	*(lista+4)





lista + 0	<i>aponta para</i>	lista[0]
lista + 1	<i>aponta para</i>	lista[1]
lista + 2	<i>aponta para</i>	lista[2]
lista + 3	<i>aponta para</i>	lista[3]
lista + 4	<i>aponta para</i>	lista[4]

• Movendo um ponteiro:

```

1. void main()
2. {
3.   int *ptr, lista[5] = {10, 20, 30, 40, 50};
4.   ptr = lista;
5.   printf("O valor do conteúdo de ptr é: %d, n);
6.   ptr = ptr + 1;
7.   printf("O valor do conteúdo de ptr é: %d, n);
8.   ptr = ptr + 3;
9.   printf("O valor do conteúdo de ptr é: %d, n);
10.  ptr = ptr - 1;
11.  printf("O valor do conteúdo de ptr é: %d, n);
12.  ptr = ptr - 1;
13.  printf("O valor do conteúdo de ptr é: %d, n);
14. }
    
```

Posições endereçadas por ptr

Linha	lista[0] 10	lista[1] 20	lista[2] 30	lista[3] 40	lista[4] 50
4					
6					
8					
10					
12			