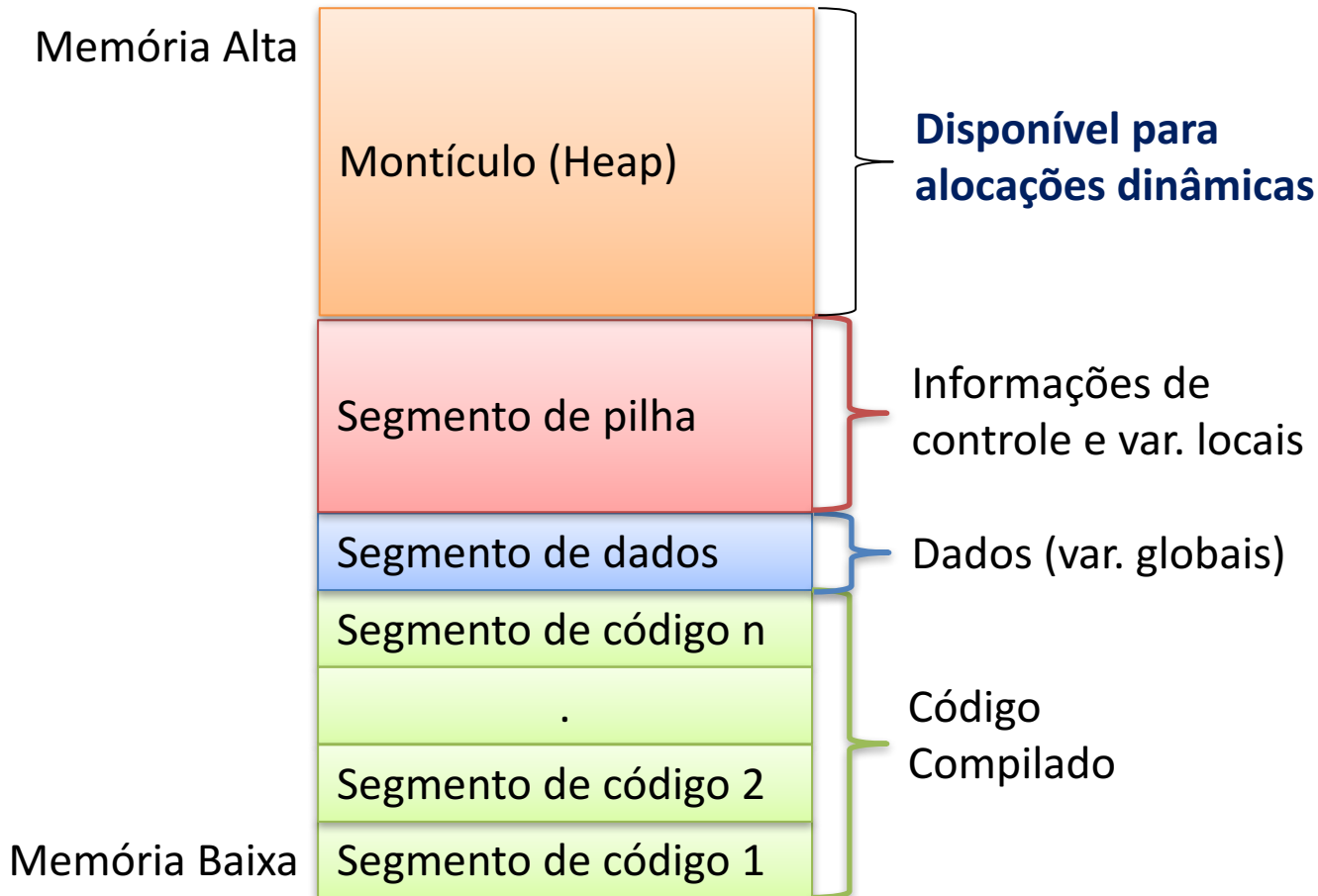


# Estrutura de Dados e Algoritmos

## Alocação Dinâmica de Memória

- Necessário sempre que a quantidade de memória requerida só é sabida em tempo de execução:
  - Carregamento dos dados de um arquivo;
  - Operações com vetores/matrizes cujas dimensões dependem do contexto do problema;
  - ...

- Mapa de memória de um programa:



- Vantagens:
  - Controle do uso de memória: uso sob demanda;
- Cuidados:
  - Gerenciar lixo;
- Implementação em C/C++:
  - Via ponteiros;
  - Funções/Operadores de alocação e desalocação;

- malloc:

```
void* malloc (size_t qtde);
```

*qtde: quantidade de bytes a serem alocados.*

tipo do dado \*nome = (tipo do dado \*) malloc(sizeof(tipo do dado) \* tamanho);

- calloc:

```
void* calloc (size_t qtde, size_t tam);
```

*qtde: quantidade de elementos a serem alocados.*

*tam: tamanho (em bytes) de cada elemento.*

tipo do dado \* nome = (tipo do dado \*) calloc (tamanho, sizeof(tipo do dado));

- `free`:

```
void free(void* ptr);
```

*ptr: ponteiro para a área de memória a ser desalocada.*

- *`#include <stdlib.h>`*

- `malloc`, `calloc`
- `free`

```
// Exemplo - C: malloc, calloc, free
#include <stdlib.h>    // Biblioteca de: malloc, calloc, free

int main ()
{
    int *Vet1, *Vet2;
    Vet1 = (int*) malloc (100*sizeof(int));
    Vet2 = (int*) calloc (100, sizeof(int));
    free (Vet1);
    free (Vet2);
    return 0;
}
```

//Declaração dos ponteiros  
//Alocação de memória  
//Alocação de memória  
//Desalocação de memória  
//Desalocação de memória

- new:

`tipoD *nome = new tipoD[qtde];`

*qtde: quantidade de elementos do tipo **tipoD** a serem alocados.*

`tipoD *nome = new tipoD[tamanho];`

- delete[]:

`void delete[];`

`delete[] Vetor;`

*Vetor: ponteiro alocado pelo comando new*

- **new e delete são implementados como operadores** da própria linguagem C++. Isto significa que não há necessidade de incluir nenhuma biblioteca ao código.

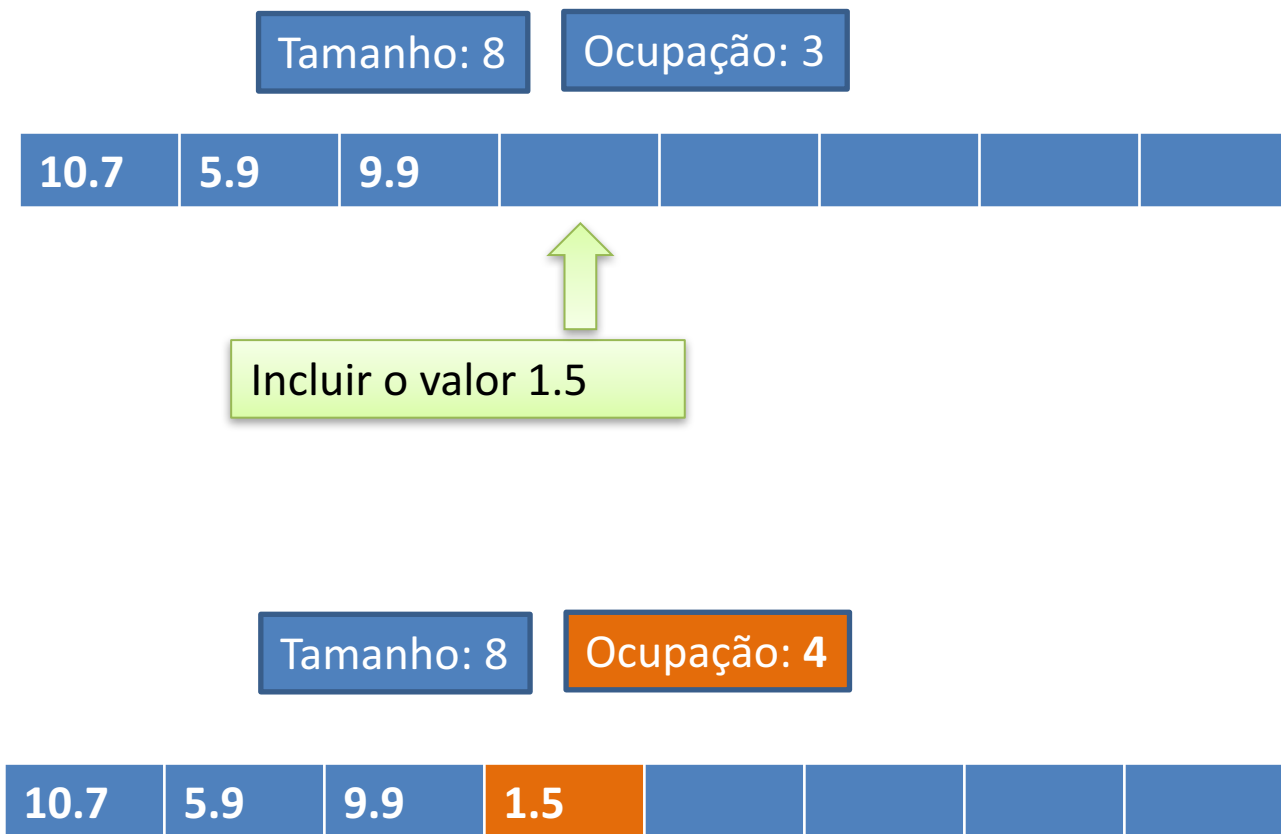


```
// Exemplo – C++: new e delete[]
int main ()
{
    int pind, ptam;
    float *pVetor;                //Declaração dos ponteiros
    printf ("Digite a qtde de elementos: ");
    scanf ("%d",ptam);
    pVetor = new float[ptam];      //Alocação de memória
    for(pind=0;pind<ptam;pind++)
        pVetor[pind] = ((float)pind)/ptam;
    for(pind=0;pind<ptam;pind++)
        printf ("%f",pVetor[pind]);
    delete[] pVetor;              //Desalocação de memória
    return 0;
}
```

- Exercícios...
  - Exercício 1;
  - Exercício 2.

- Um vetor, seja alocado estaticamente, seja alocado dinamicamente, não guarda informações sobre o seu tamanho;
- Sempre que criar funções que recebem vetores como parâmetros de entrada, lembre-se de passar o tamanho do vetor também.

- Inclusão:



- Exclusão:

Tamanho: 8

Ocupação: 4

10.7	5.9	9.9	1.5				
------	-----	-----	-----	--	--	--	--



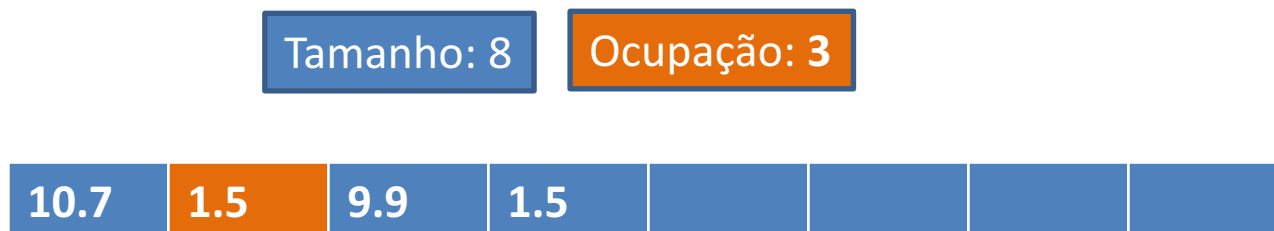
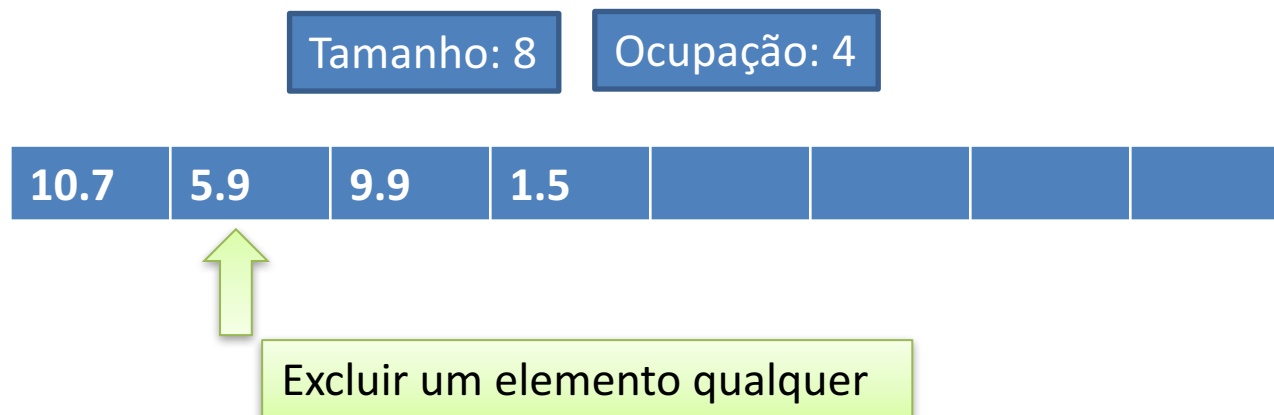
Excluir o último elemento

Tamanho: 8

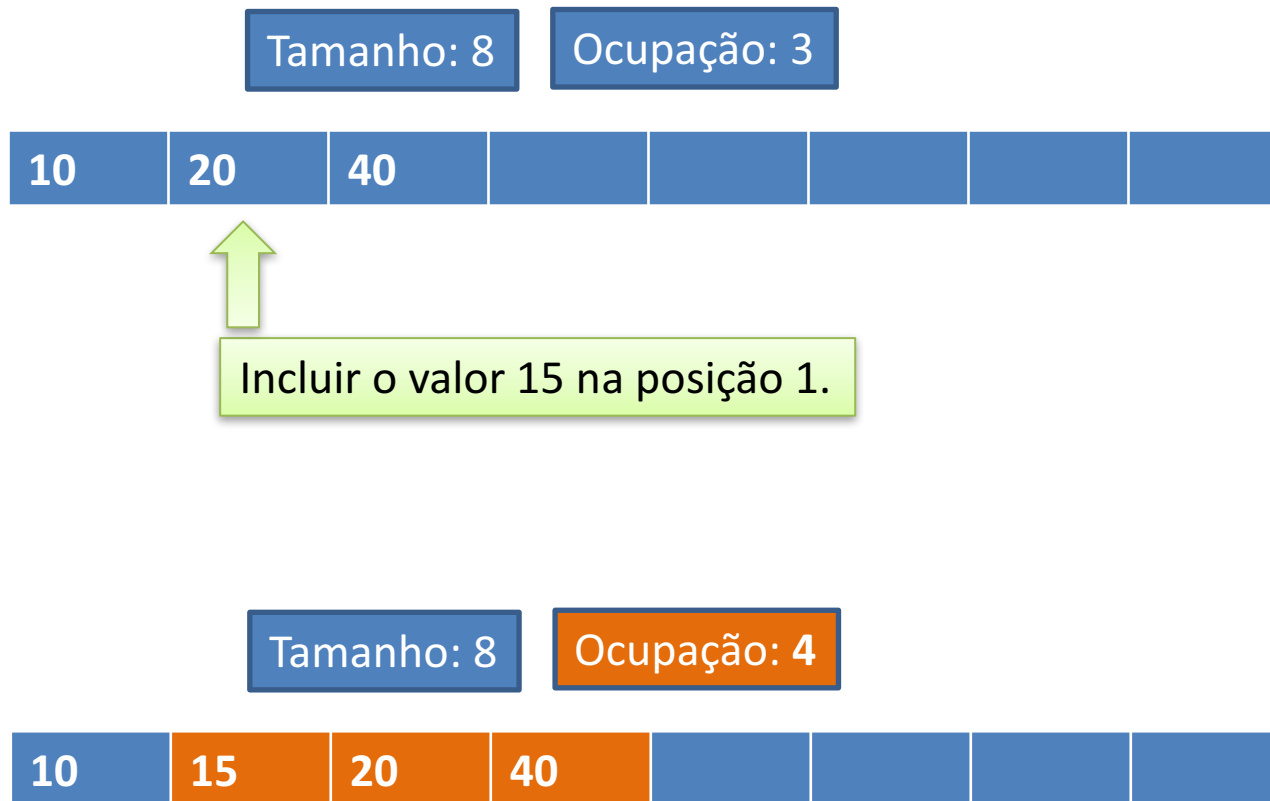
Ocupação: 3

10.7	5.9	9.9	1.5				
------	-----	-----	-----	--	--	--	--

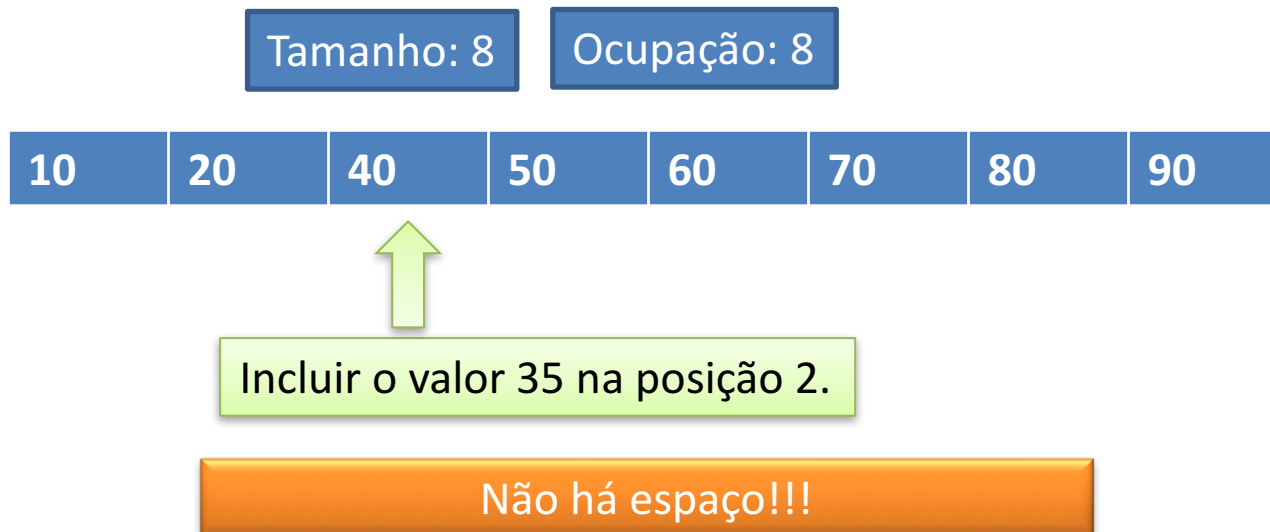
- Exclusão:



- Inclusão:

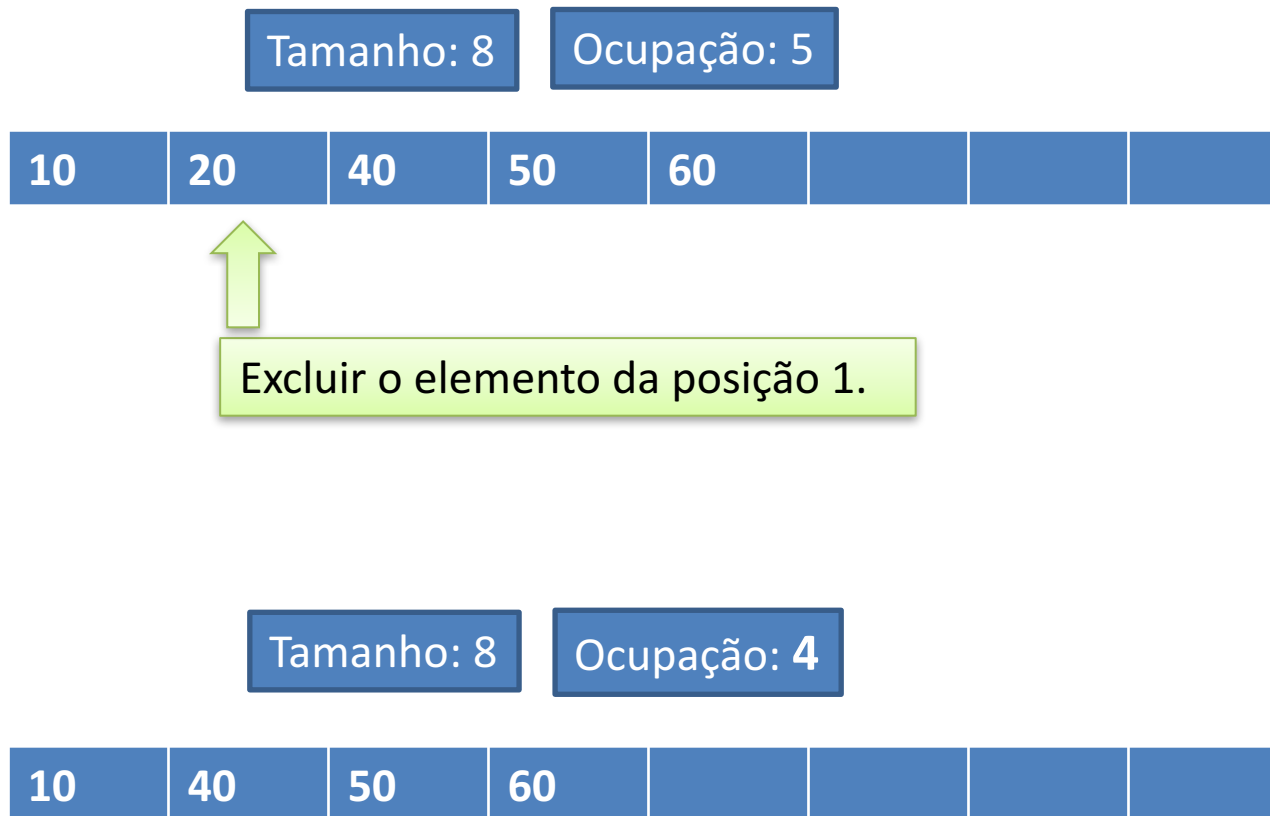


- Inclusão:





- Exclusão:



# Prática...

- No ambiente virtual (Moodle):
  - Prática Vetores Dinâmicos.pdf
  - Arquivo de dados: notas.txt