Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

LISTAS, PILHAS E FILAS

2011-12-12

2011/2012, A1, S1

PAULO NUNES

AV. DR. FRANCISCO SÁ CARNEIRO, 50 - 6301-559 GUARDA

TELF. 271220161, EXT. 161, GAB:20

GPS: LATITUDE: 40.5416236730513, LONGITUDE: -7.28243350982666

VOIP: pnunes@ipg.pt, MSN: pnunes@ipg.pt, SKYPE: pnunes.ipg.pt

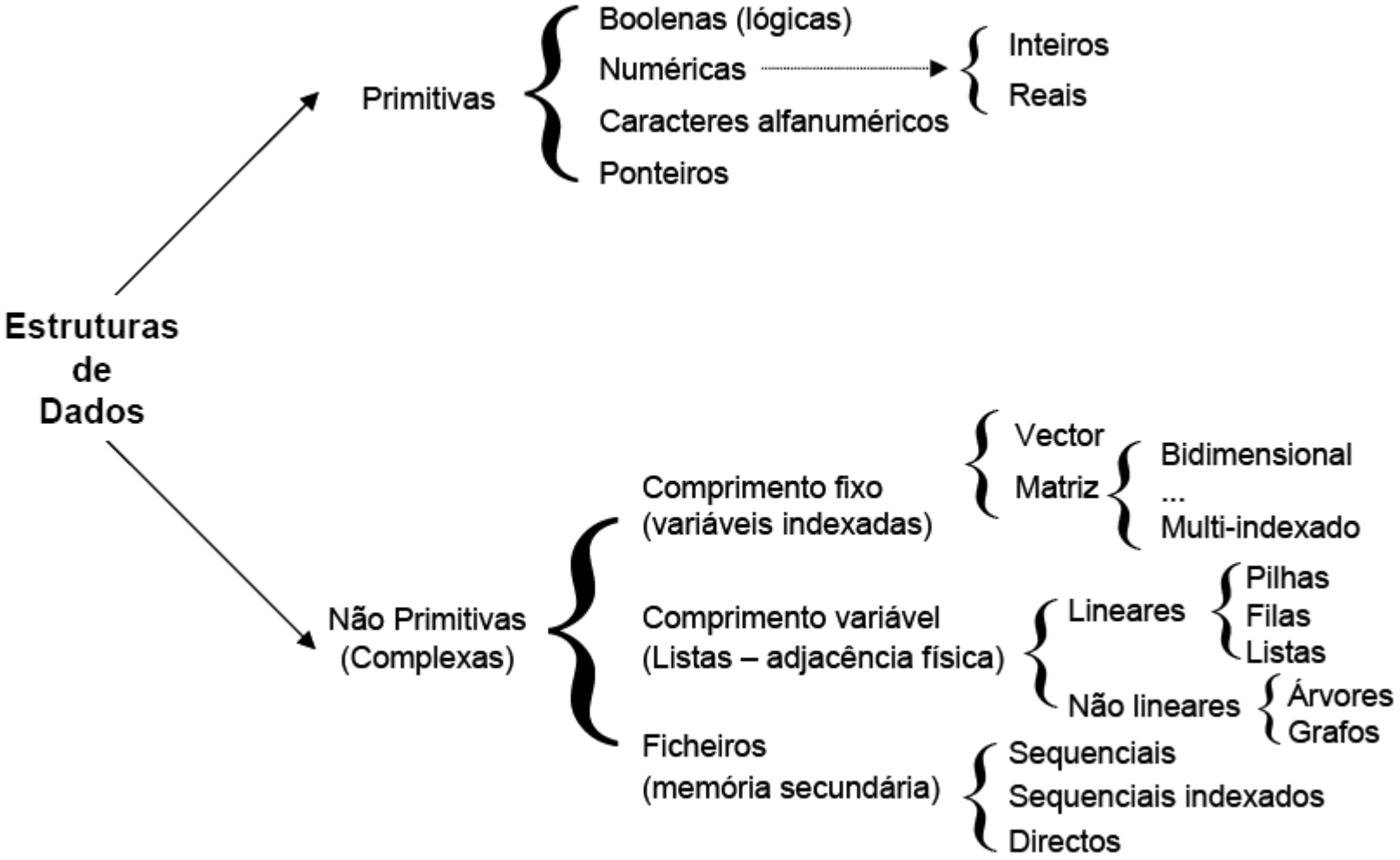
EMAIL: Mailto:pnunes@ipg.pt, WEB: http://www.ipg.pt/user/~pnunes/

SUMÁRIO

- Estruturas de dados
  - Visão geral
- Vetores
  - Vantagens
  - Desvantagens
- Listas
  - Definição e exemplos
  - Aplicações
  - Operações e Complexidade
  - Estrutura física

ESTRUTURAS DE DADOS

Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda



VETORES

- Vantagens
  - Permite a manipulação como um todo:
    - Gravar, recuperar, copiar.
- Desvantagens
  - O tamanho tem de ser conhecido.
  - Inserir de um novo elemento no vetor obriga a movimentar todos os restantes para a direita de uma posição.
  - A eliminação de um elemento no vetor obriga a movimentar todos os restantes para a esquerda de uma posição.

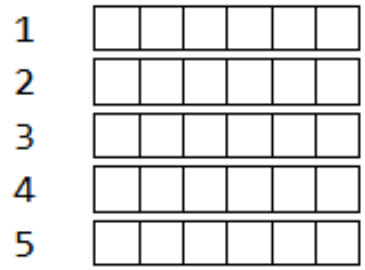
VECTOR – NÍVEL FÍSICO

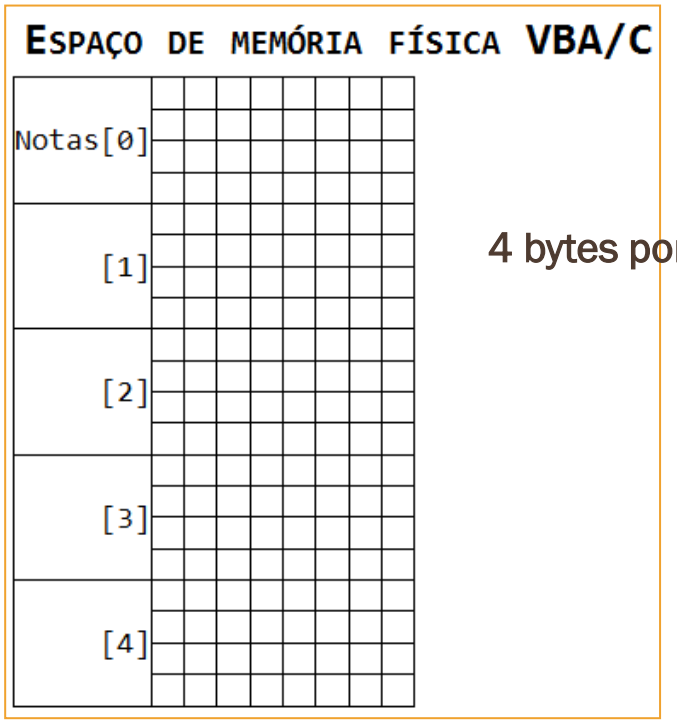
Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

- Os dados de um vetor estão todos seguidos e a uma distância igual.

Memória Lógica

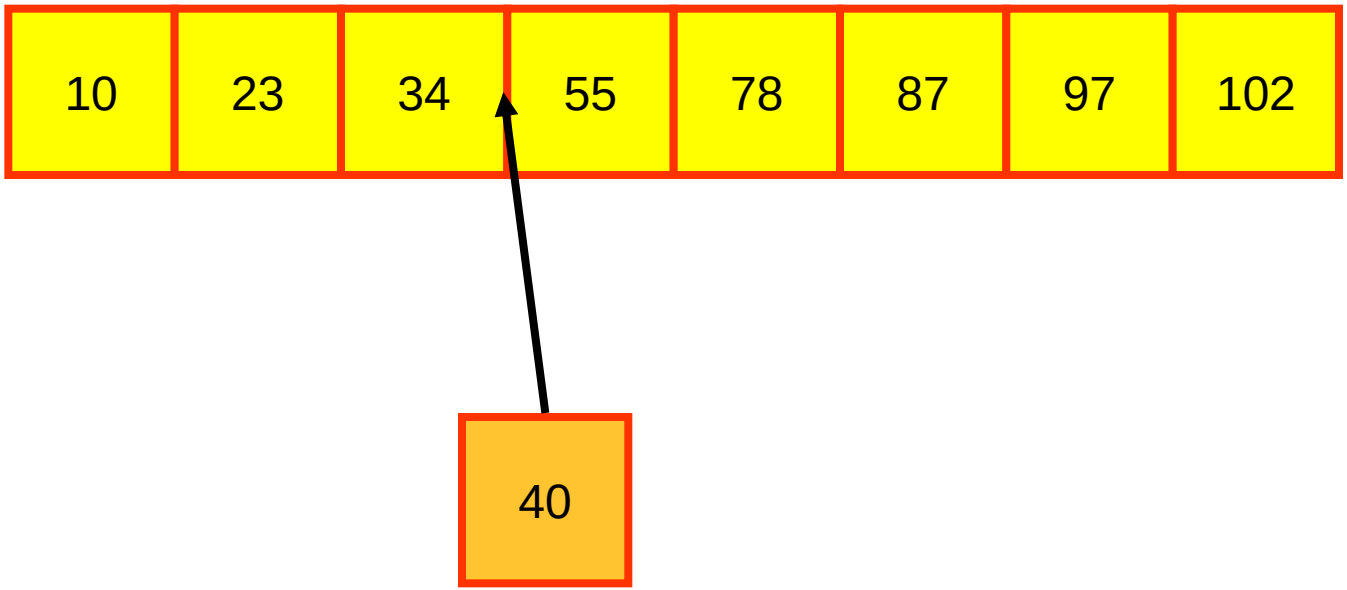
Notas (Inteiro T6) [5] (>= Li0, <= Ls0)



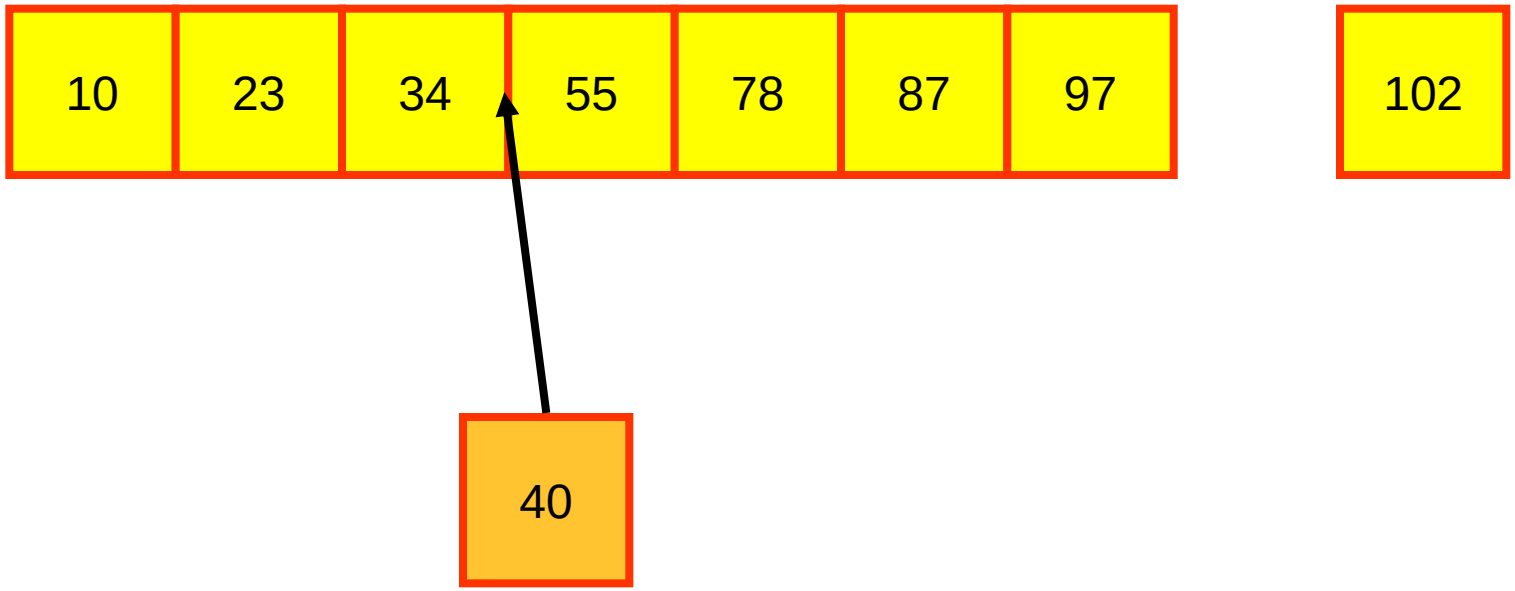


4 bytes por item.

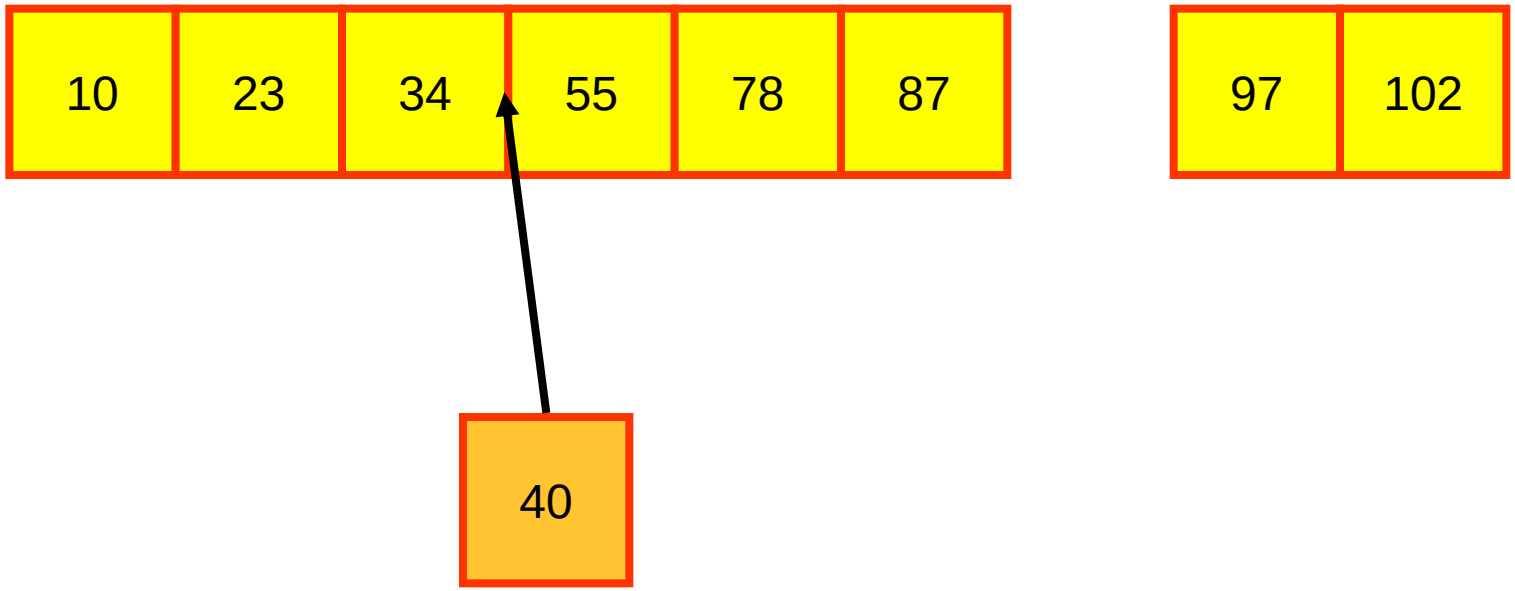
VECTOR:INSERIR



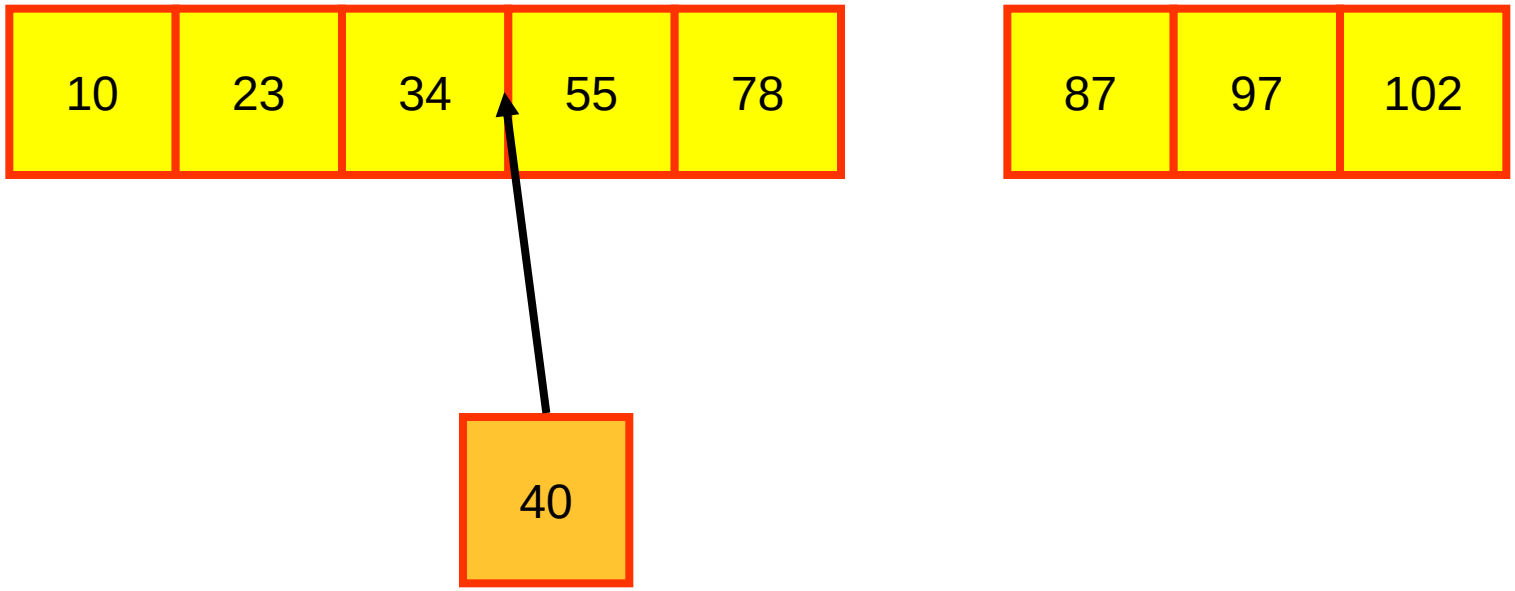
VECTOR:INSERIR



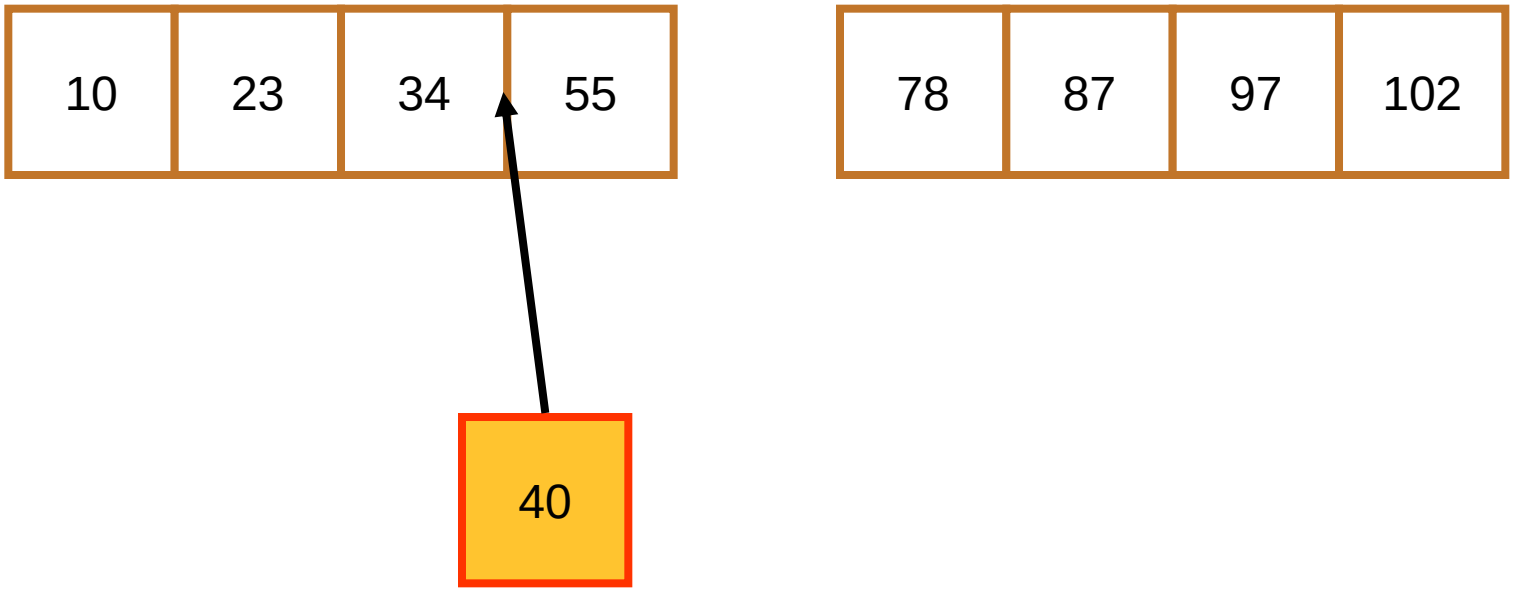
VECTOR:INSERIR



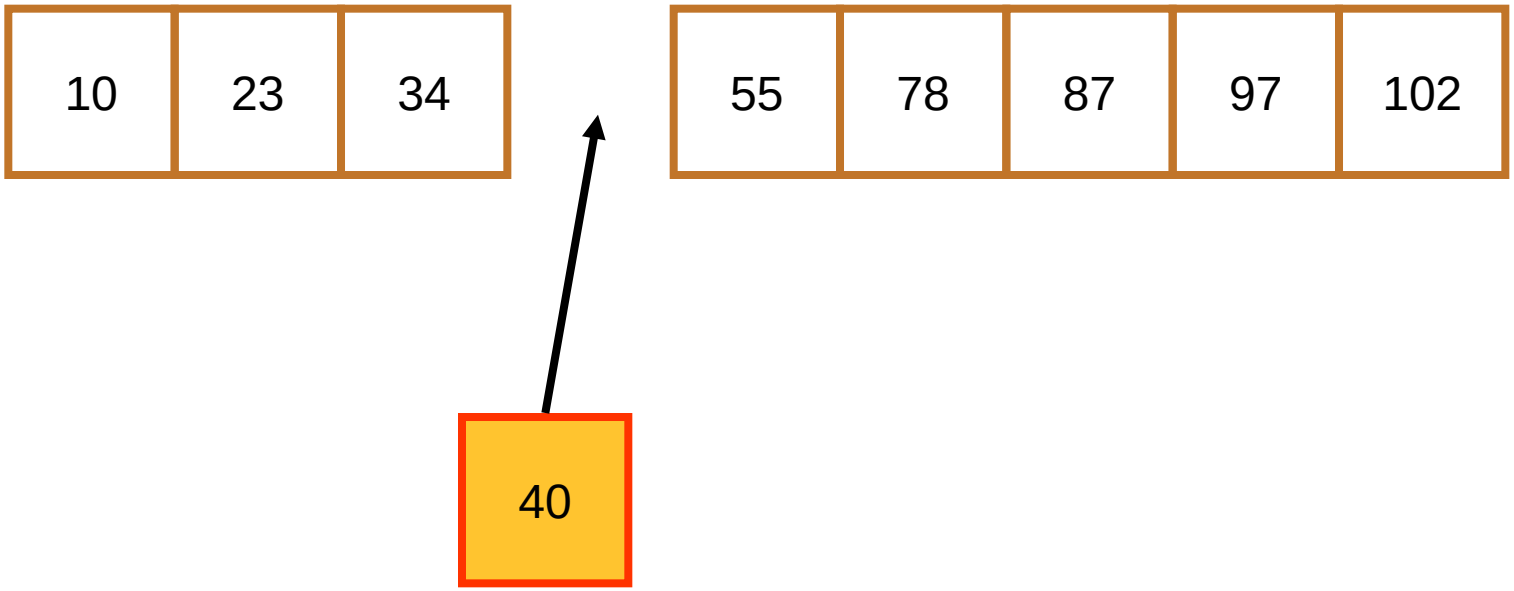
VECTOR:INSERIR



VECTOR:INSERIR



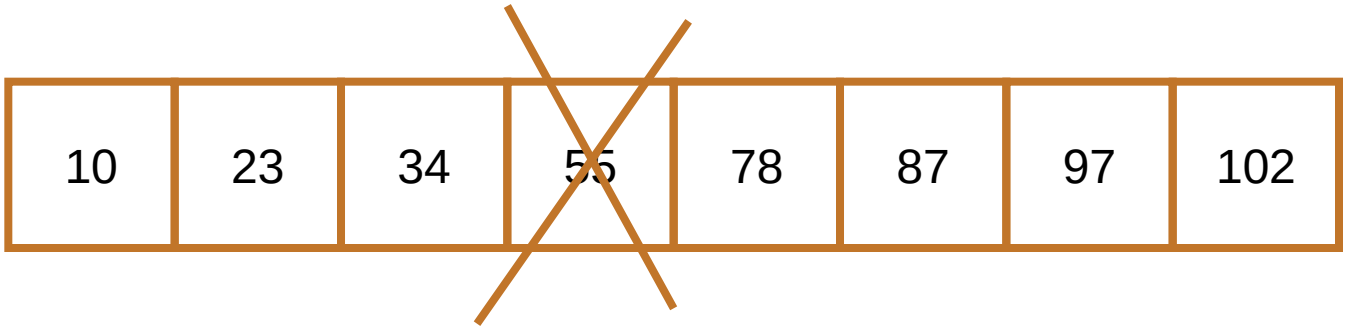
VECTOR:INSERIR



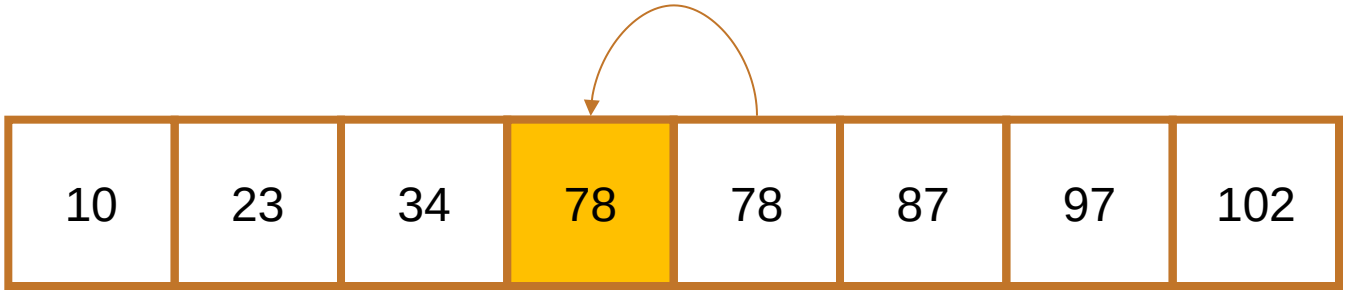
VECTOR:INSERIR

10	23	34	40	55	78	87	97	102

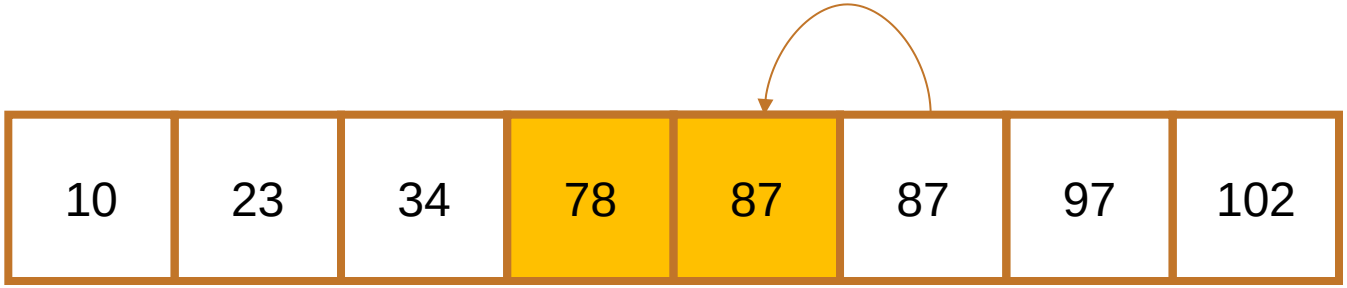
VECTOR:ELIMINAR



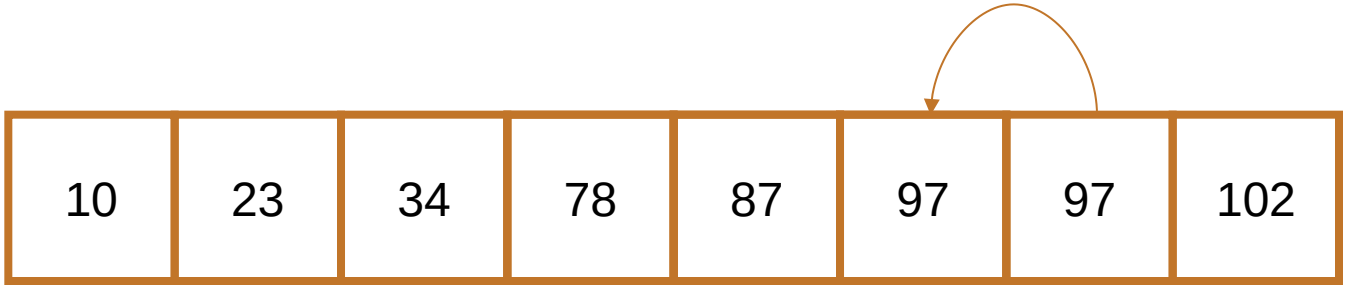
VECTOR:ELIMINAR



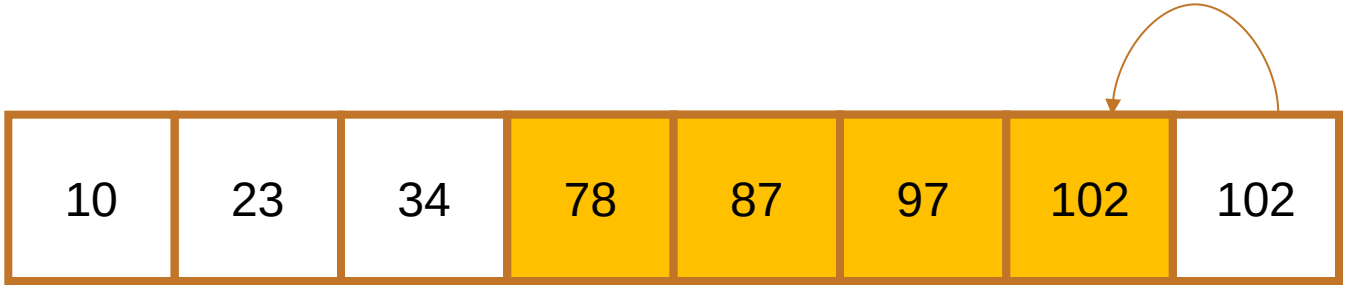
VECTOR:ELIMINAR



VECTOR:ELIMINAR



VECTOR:ELIMINAR



VECTOR:ELIMINAR

10	23	34	78	87	97	102

LISTA

- É uma coleção de nós, em que cada nó contém um conjunto de atributos, entre os quais um atributo ***proximo***
  - que contém uma referência para o nó que ocupa a próxima posição na lista.
- Os restantes atributos contêm informação em função da aplicação.
  - O acesso à lista é realizado através de uma referência especial denominada de ***inicio***
    - que contém uma referência para o primeiro elemento da lista.

LISTAS: APLICAÇÃO

- As listas ligadas são estruturas de dados fundamentais.
- Podem funcionar como componentes básicos ***building blocks*** para implementar outras estruturas de dados tais como:
  - filas
  - pilhas
  - vetores associativos

OPERAÇÕES BÁSICAS

- As operações básicas que podemos efetuar numa lista são:
  - inserção
  - eliminação
  - consulta de nós.
- Os nós podem ser inseridos ou eliminados dinamicamente em qualquer posição da lista.

EXEMPLO: INSERÇÃO E ELIMINAÇÃO

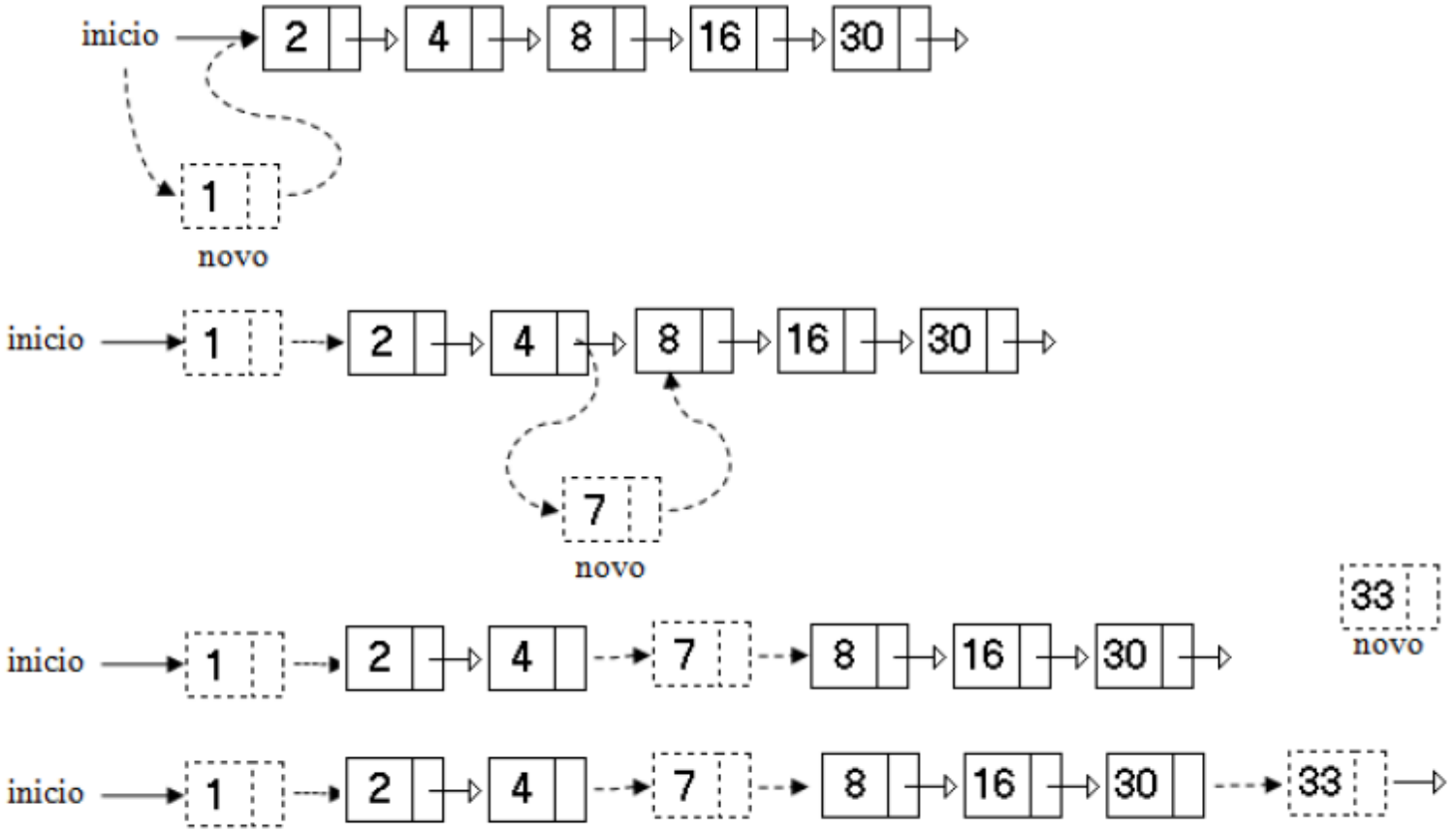


Figura 13.1: Operações de inserção/eliminação de elementos numa lista ligada.

INSERÇÃO

- A inserção de elementos numa lista pode ocorrer em três posições:
  - no início
  - no meio
  - no fim.

ELIMINAÇÃO

- As operações de eliminação pode ocorrer nas mesmas posições da inserção:
  - no início
  - no meio
  - no fim.
- Para a remoção de elementos da lista são realizadas as operações inversas às da inserção de elementos.

COMPLEXIDADE DAS OPERAÇÕES

	Média	Pior caso
Espaço	O(n)	O(n)
Pesquisa	O(n/2)	O(n)
Inserção	O(n/2)	O(n)
Eliminação	O(n/2)	O(n)

Tabela 13.1: Complexidade das operações numa lista.

ESTRUTURA FÍSICA

- Os nós podem estar em qualquer parte da memória
- Cada nó tem apenas um endereço para o próximo nó da lista.
  - Que permite a passagem de um nó para o próximo.

LISTA DE INTEIROS

- Estrutura: ListaE
  - Numero (INTEIRO) – 1 byte
  - Endereco_proximo (INTEIRO) – 1 byte

- Um endereço é um número que representa a posição inicial de memória de um conjunto de informação.

ESPAÇO FÍSICO

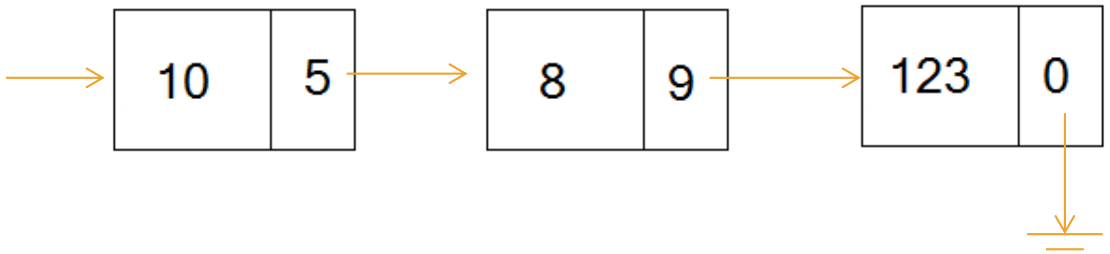
Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

Bytes

ESPAÇO DE MEMÓRIA FÍSICA

M[0]								
[1]								
[2]								
[3]	0	0	0	0	1	0	1	0
[4]	0	0	0	0	0	1	0	1
[5]	0	0	0	0	1	0	0	0
[6]	0	0	0	0	1	0	0	1
[7]								
[8]								
[9]	0	1	1	1	1	0	1	1
[10]	0	0	0	0	0	0	0	0
[11]								
[12]								
[13]								
...								
[1023]								
inicio	0	0	0	0	0	0	1	1

Que elementos estão na memória ?
10, 5, 8, 9, 123,0



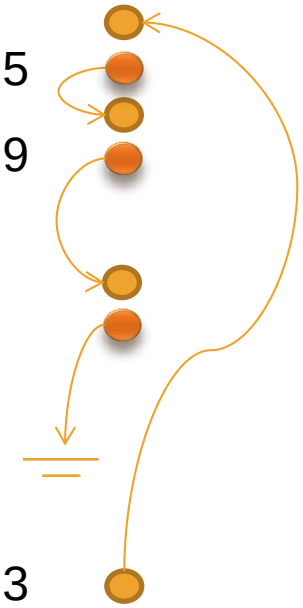
ESPAÇO FÍSICO

Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

Bytes

ESPAÇO DE MEMÓRIA FÍSICA

M[0]									
[1]									
[2]									
[3]	0	0	0	0	1	0	1	0	
[4]	0	0	0	0	0	1	0	1	5
[5]	0	0	0	0	1	0	0	0	
[6]	0	0	0	0	1	0	0	1	9
[7]									
[8]									
[9]	0	1	1	1	1	0	1	1	
[10]	0	0	0	0	0	0	0	0	
[11]									
[12]									
[13]									
...									
[1023]									
inicio	0	0	0	0	0	0	1	1	3



Que elementos estão na memória ?
10, 5, 8, 9, 123,0

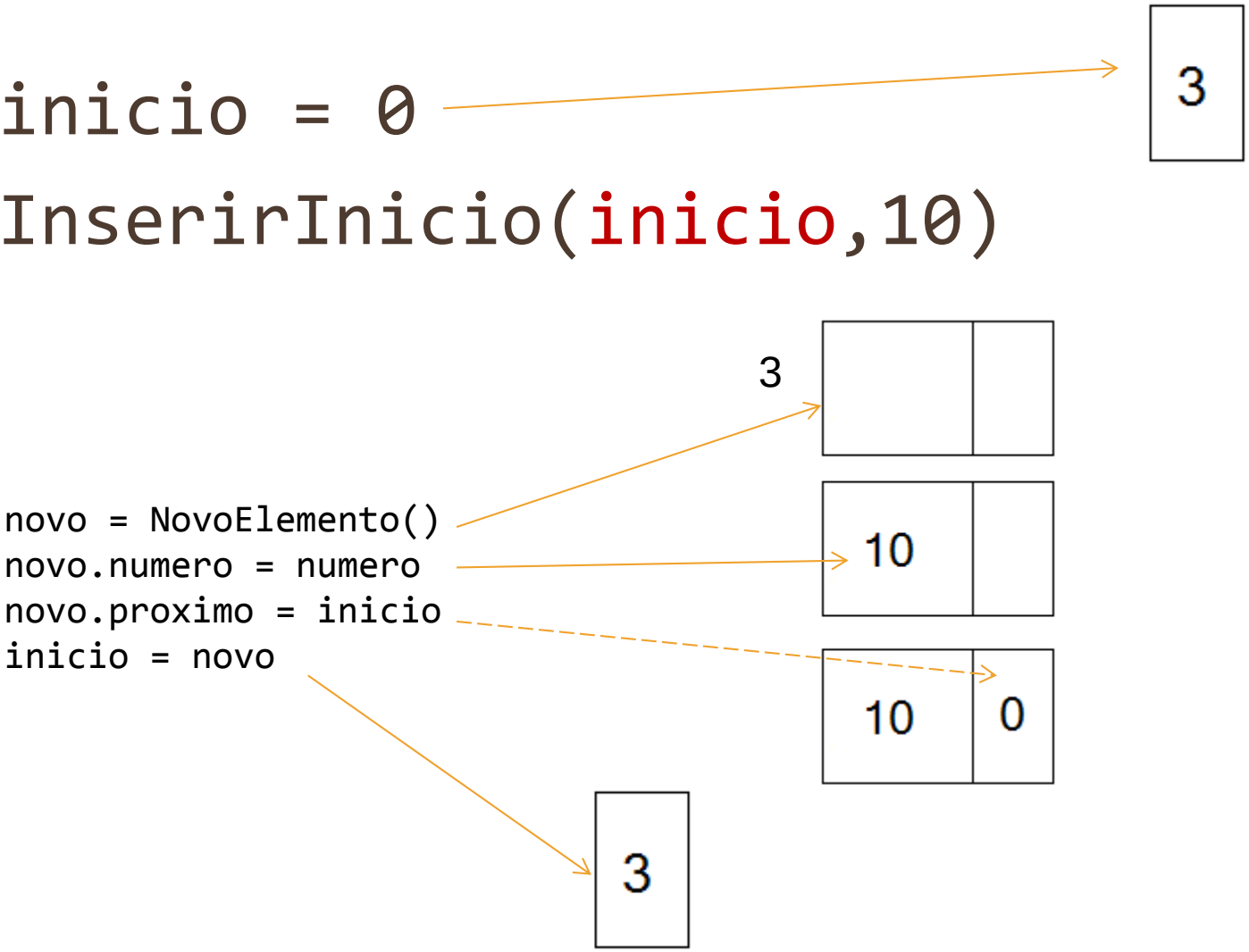
O endereço zero está reservado para o Sistema Operativo.
Este é utilizado para indicar que a lista não tem mais elementos.

INSERIR ELEMENTO


```
InserirInicio(inicio (endereço),
              numero (Inteiro))
  Início:
     novo = NovoElemento()
     novo.numero = numero
     novo.proximo = inicio
     inicio = novo;
   Fim.
```

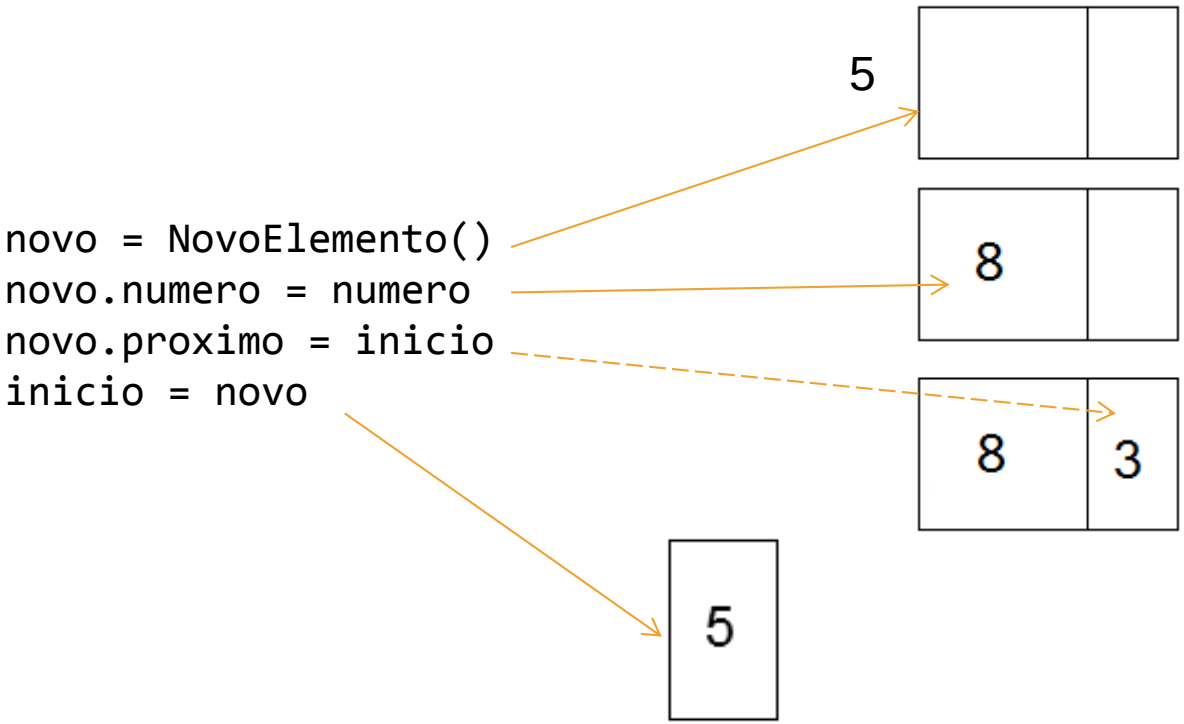
LISTA VAZIA: E10


```
inicio = 0
InserirInicio(inicio,10)
```



LISTA: E8

InserirInicio(inicio,8)



ELIMINAR ELEMENTO


```
EliminarInicio(inicio (endereço))
  Início:
     Se (inicio) Então
          aux = inicio
          inicio = inicio.proximo
          EliminaElemento aux
     FimSe
   Fim.
```

INSERIR ELEMENTO FIM

Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

```
InserirFim(inicio (endereço), numero (Inteiro))
    Início:
          novo = NovoElemento()
          novo.numero = numero
          novo.proximo = 0
          ant = 0
          aux = inicio
          Enquanto (aux)                    /* ant referencia o último */
                  ant = aux
                  aux = aux.proximo
          FimEnquanto
          Se (ant) Então                    /* lista tem pelo menos 1 elemento */
            ant.proximo = novo
          Senão                             /* lista vazia */
            inicio = novo
          FimSe
    Fim.
```

ELIMINAR ELEMENTO FIM


```
EliminaFim(inicio (endereço))
    Início:
        Se (inicio) Então
            Se (inicio.proximo)
                ap = inicio
                aux = inicio.proximo
                Enquanto (aux.proximo)    /* ap referencia o penúltimo */
                    ant = aux
                    aux = aux.proximo
                FimEnquanto
                aux = ant
            Senão                         /* só tem um */
                aux = inicio
                inicio = 0
            FimSe
            EliminaElemento aux
        FimSe
    Fim.
```

LISTA: INICIO, FIM

- Para facilitar as operações de inserção no fim da lista recorre-se a outra referência especial denominada de ***fim***
  - que contém uma referência para o último elemento da lista.

INSERIR ELEMENTO INICIO


```
InserirInicio(inicio (endereço),
                    fim (endereço),
                    numero (Inteiro))
   Início:
      novo = NovoElemento()
      novo.numero = numero
      novo.proximo = inicio
      inicio = novo
      Se (fim = 0) Então
         fim = inicio
      FimSe
    Fim.
```

INSERIR ELEMENTO FIM


```
InserirFim(inicio (endereço),
                  fim (endereço),
                  numero (Inteiro))
   Início:
       novo = NovoElemento()
       novo.numero = numero
       novo.proximo = 0
       Se (fim) Então
           fim.proximo = novo
           fim = novo
       Senão
           inicio = novo
           fim = novo
       fimSe
    Fim.
```

PESQUISA ELEMENTO


```
ExisteElemento(inicio (endereço),
                    fim (endereço),
                    e (Inteiro))
   Início:
        aux = inicio
        enc = 0
        Enquanto (aux)
          Se (aux.numero = numero) Então
             enc = 1
             aux = 0
          FimSe
          aux = aux.proximo
        FimEnquanto
        RETORNA enc
    Fim.
```

FILA

- Uma fila é uma lista ligada na qual a colocação e retirada de elementos têm lugar em opostos (início e fim).
- Nas filas temos duas operações básicas, colocar e retirar elementos.
- As filas seguem o método de FIFO, que significa "first-in-first-out", ou seja o primeiro elemento a ser colocado na fila é o primeiro elemento a ser retirado.

FILA: OPERAÇÕES

- Colocar elemento na fila
  - InserirFim()
    - ColocaElemento
- Retirar elemento da fila
  - RetiraInicio()
    - RetiraElemento

COLOCA ELEMENTO FILA


```
ColocaElemento(inicio (endereço),
               fim (endereço),
               numero (Inteiro))

Início:
     InserirFim(inicio,fim,numero)
Fim.
```

RETIRA ELEMENTO INÍCIO


```
RetiraElemento(inicio (endereço),
                    fim (endereço),
                    numero (Inteiro))
   Início:
      Se (inicio) Então
         aux = inicio
         inicio = inicio.proximo
         Se (inicio = 0) Então
            fim =0
         FimSe
         RETORNA aux
      FimSe
      RETORNA 0
    Fim.
```

PILHA

- Uma pilha é uma lista ligada na qual a colocação e retirada de elementos têm lugar na mesmo posição (início).
- Nas pilhas temos duas operações básicas, colocar e retirar elementos.
- As pilhas seguem o método de LIFO, que significa "last-in-first-out", ou seja o último elemento a ser colocado na pilha é o primeiro elemento a ser retirado.

PILHA: OPERAÇÕES

- Colocar elemento na pilha
  - InserirInicio ()
    - ColocaElemento
- Retirar elemento da pilha
  - RetiraInicio()
    - RetiraElemento

COLOCA ELEMENTO PILHA


```
ColocaElemento(inicio (endereço),
               fim (endereço),
               numero (Inteiro))

Início:
     InserirInicio(inicio,fim,numero)
Fim.
```

RETIRA ELEMENTO INÍCIO


```
RetiraElemento(inicio (endereço),
                    fim (endereço),
                    numero (Inteiro))
   Início:
      Se (inicio) Então
         aux = inicio
         inicio = inicio.proximo
         Se (inicio = 0) Então
            fim =0
         FimSe
         RETORNA aux
      FimSe
      RETORNA 0
    Fim.
```

BIBLIOGRAFIA

- Knuth, Donald E. (1993). The Art of Computer Programming – VOLUME 3 -Sorting and Searching. Third Edition, Prentice Hall.