Aula 04 - Conteúdo

1) Listas Ligadas e suas representações
2) Algoritmos de lista ligada – utilizando alocação dinâmica e vetores
3) Exercícios

Listas

Uma lista é uma coleção ordenada de componentes do mesmo tipo (registro). Por exemplo: catálogo telefônico, cartas do baralho, alunos do curso de EDA etc.

Operações efetuadas numa lista

- inserir um elemento (em qualquer posição)
- remover um lemento (de qualquer posição)
- acessar um elemento da lista (consultar ou alterar)
- verificar se a lista está vazia ou cheia

Lista Seqüencial

Para cada componente da lista, o sucessor deste componente está armazenado na posição física seguinte da lista. Normalmente este tipo de lista é implementado com vetores.

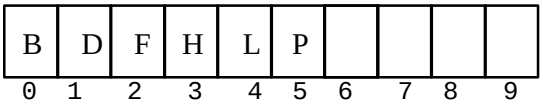
Vantagens

- rapidez nas operações de acesso (consulta, alteração, impressão)
- algoritmos simples

Desvantagens

- operações de inserção e deleção provocam movimentação dos dados

Exemplo:
Inserir o elemento C na lista
Deletar o elemento F da lista
Imprimir o elemento da posição 3



Lista Ligada

Coleção ordenada de elementos (registros) em que cada um deles possui uma referência (campo) indicando a localização do próximo componente da lista. A ordenação da lista é dada explicitamente por esta referência (campo).

Vantagens

- operações de inserção e remoção não exigem movimentação de dados

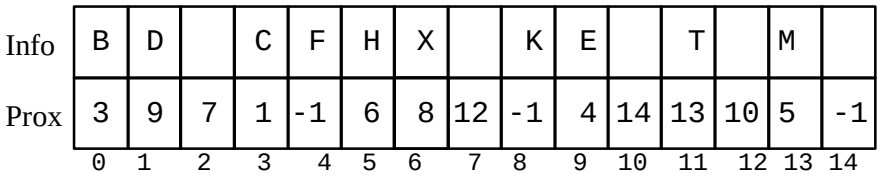
Desvantagens

- para acessar um elemento deve-se percorrer a lista
- algoritmos mais complexos se comparados com a lista seqüencial

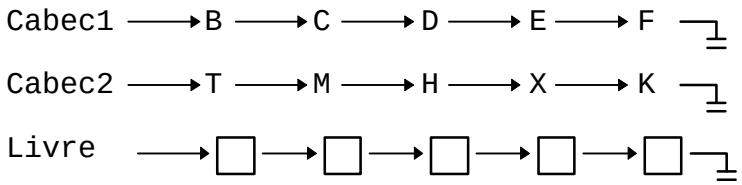
Representações de listas ligadas

Estudaremos 2 maneiras de representar lista ligada: uma utilizando vetor e a outra utilizando alocação dinâmica. Ambas implementações utilizam apontadores. A implementação com vetor utiliza como apontador o próprio índice do mesmo e a implementação com alocação dinâmica utiliza endereços de memória.

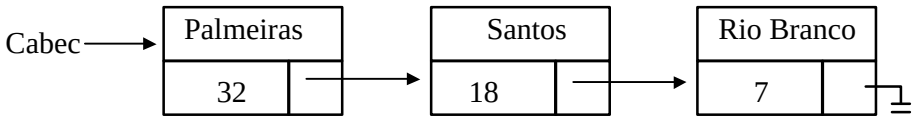
Esquema de implementação com vetor



Cabec1 = 0 Cabec2 = 11 Livre = 2

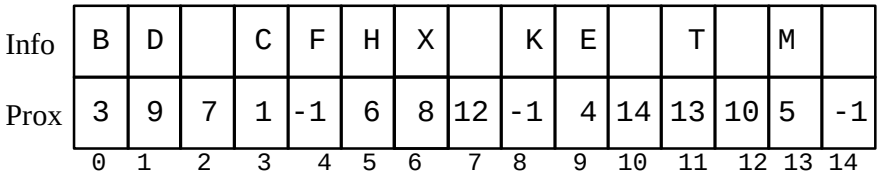


Esquema de implementação com alocação dinâmica

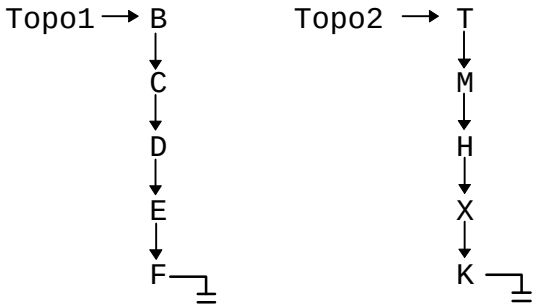


Algoritmos - implementações com vetor e com alocação dinâmica

Exemplo 4.1 – Pilha utilizando estrutura com vetor (ProjEx401.dpr e Ex401.pas)

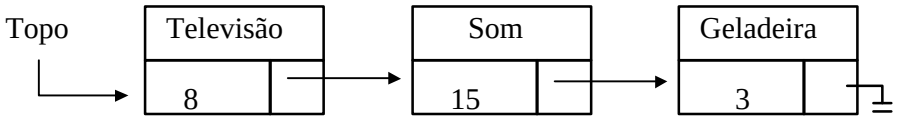


Topo1 = 0 Topo2 = 11 Livre = 2



```
Convenções:
Topo    = -1 indica pilha vazia
Livre   = -1 indica pilha cheia
TAM_VET =  15

INICIALIZAÇÃO
Topo = -1
Livre = 0
Para  i = 0  até TAM_VET-1 faça  P[i].Prox = i + 1
P[TAM_VET-1].Prox = -1

INSERÇÃO
Se Livre = -1  então  "INSERÇÃO IMPOSSÍVEL"
senão
  atual = Livre
  Livre = P[Livre].Prox
  P[atual].Info = Elem
  P[atual].Prox = Topo
  Topo = atual
fimse

DELEÇÃO
Se Topo = -1 então "DELEÇÃO IMPOSSÍVEL"
senão
  atual = Topo
  Elem  = P[Topo].Info
  Topo  = P[Topo].Prox
  P[atual].Prox = Livre
  Livre = atual
fimse
```

Exemplo 4.2 – Pilha utilizando alocação dinâmica (ProjEx402.dpr e Ex402.pas)

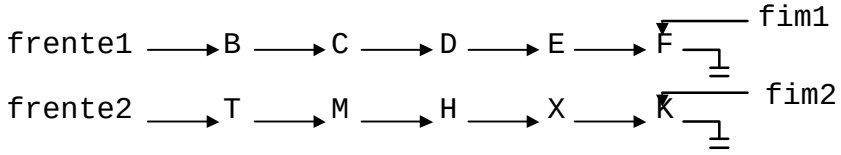



```
Convenções:
Topo = NULL indica pilha vazia

INICIALIZAÇÃO
Topo = NULL

INSERÇÃO
E = Alocar espaço
E->Info = Elemento
E->Prox = Topo
Topo = E

REMOÇÃO
Se Topo = NULL então "REMOÇÃO IMPOSSIVEL"
senão
  Elem = Topo->Info
  E = Topo
  Topo = Topo->Prox
  Liberar espaço(E)
Fimse
```

Exemplo 4.3 – Fila utilizando estrutura com vetor (ProjEx403.dpr e Ex403.pas)

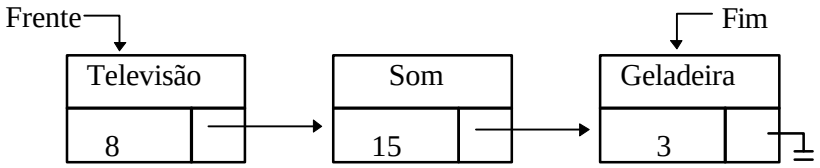
Info	B	D		C	F	H	X		K	E		T		M	
Prox	3	9	7	1	-1	6	8	12	-1	4	14	13	10	5	-1
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

frente1 = 0 fim1 = 4 frente2 = 11 fim2 = 8 Livre = 2




```
Convenções:
fim = frente = -1 indica fila vazia
Livre = -1 indica fila cheia
TAM_VET =  15

INICIALIZAÇÃO
frente = -1
fim    = -1
Livre  = 0
Para  i = 0  até  TAM_VET-1 faça  F[i].Prox = i + 1
F[TAM_VET-1].Prox = -1


INSERÇÃO
Se Livre = -1 então "INSERÇÃO IMPOSSÍVEL"
senão
  atual  =  Livre
  Livre  =  F[Livre].Prox
  F[atual].Info = Elem
  F[atual].Prox = -1
  Se fim = -1 então
    frente = fim = atual
  senão
    F[fim].Prox = atual
    fim = atual
  fimse
fimse

DELEÇÃO
Se frente = -1 então "DELEÇÃO IMPOSSÍVEL"
senão
  atual = frente
  Elem  = F[frente].Info
  frente = F[frente].Prox
  Se frente = -1 então Fim = -1
  F[atual].Prox = Livre
  Livre = atual
fimse
```

Exemplo 4.4 – Fila utilizando alocação dinâmica (ProjEx404.dpr e Ex404.pas)

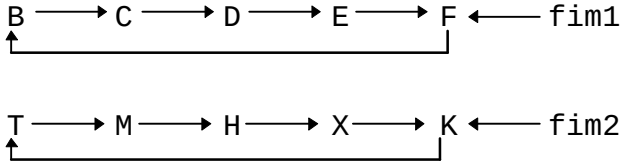



```
Convenções:
Frente = Fim = NULL indica fila vazia

INICIALIZAÇÃO
Frente = Fim = NULL

INSERÇÃO
E = Alocar Espaço
E->Info = Elemento
E->Prox = NULL
Se Fim = NULL então
  Fim = Frente = E
senão
  Fim->Prox = E
  Fim = E
fimse

REMOÇÃO
Se Frente = NULL então "REMOÇÃO IMPOSSIVEL"
senão
  E = Frente
  Elemento = E->Info
  Frente = E->Prox
  liberar espaço(E)
  Se Frente = NULL então
    Fim = NULL
  fimse
fimse
```

Exemplo 4.5 - Fila Circular utilizando estrutura com vetor (ProjEx405.dpr e Ex405.pas)

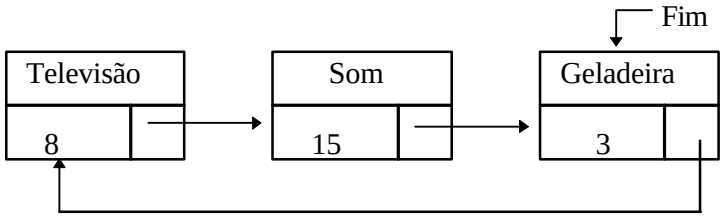
Info	B	D		C	F	H	X		K	E		T		M	
Prox	3	9	7	1	0	6	8	12	11	4	14	13	10	5	-1
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

fim1 = 4 fim2 = 8 Livre = 2



```
Convenções:
fim = frente = -1 indica fila vazia
Livre = -1 indica fila cheia
TAM_VET =  15


INICIALIZAÇÃO
fim    = -1
Livre  = 0
Para  i = 0  até  TAM_VET-1 faça  FC[i].Prox = i + 1
FC[TAM_VET-1].Prox = -1

INSERÇÃO
Se Livre = -1 então "INSERÇÃO IMPOSSÍVEL"
senão
  atual  =  Livre
  Livre  =  FC[Livre].Prox
  FC[atual].Info = Elem
  Se fim = -1 então
    FC[atual].Prox = atual
  senão
    FC[atual].Prox = FC[fim].Prox
    FC[fim].Prox = atual
  fimse
  fim = atual
fimse

DELEÇÃO
Se fim = -1 então"DELEÇÃO IMPOSSÍVEL"
senão
  atual = FC[Fim].Prox
  Elem = FC[atual].Info
  Se (atual = fim)
    fim = -1
  senão
    FC[Fim].Prox = FC[atual].Prox
  fimse
  FC[atual].Prox = Livre
  Livre  =  atual
fimse
```

Exemplo 4.6 - Fila Circular utilizando alocação dinâmica (ProjEx406.dpr e Ex406.pas)



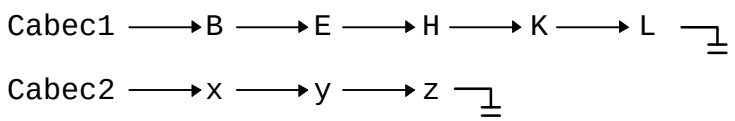

```
Convenções:
Fim = NULL indica fila vazia

INICIALIZAÇÃO
Fim = NULL
```

```
INSERÇÃO
E = Alocar Espaço
E->Info = Elemento
Se Fim = NULL então
  Fim = E
  Fim->Prox = E
senão
  E->Prox = Fim->Prox
  Fim->Prox = E
  Fim = E
fimse

REMOÇÃO
Se Fim = NULL então "REMOÇÃO IMPOSSÍVEL"
senão
  E = Fim->Prox
  Elemento = E->Info
  Fim->Prox = E->Prox
  Se Fim = E então
    Fim = NULL
  fimse
  Liberar espaço(E)
fimse
```

Exemplo 4.7 - Lista Ordenada utilizando estrutura com vetor (ProjEx407.dpr e Ex407.pas)

Info	B	H		E	L		x		z	K		y			
Prox	3	9	7	1	-1	-1	11	12	-1	4	14	8	10	5	13
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

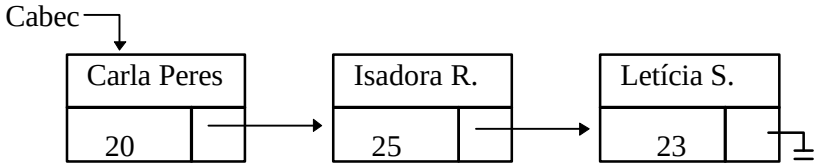
Cabec1 = 0 Cabec2 = 6 Livre = 2




```
Convenções:
Cabec = -1 indica lista vazia
Livre = -1 indica lista cheia
TAM_VET =  15

INICIALIZAÇÃO
Cabec = -1     Livre = 0
Para  i = 0 até TAM_VET-1 faça LISTA[i].Prox = i+1
LISTA[TAM_VET-1] = -1

INSERÇÃO
Se  Livre = -1 então "INSERÇÃO IMPOSSÍVEL"
senão
  Se Cabec = -1 então
    Cabec = Livre
    LISTA[Cabec].Info = Elemento
    Livre = LISTA[Livre].Prox
    LISTA[Cabec].Prox = -1
```

```
  senão
    aux = Cabec
    ant  =  -1
    Enquanto aux != -1 e LISTA[aux].Info < Elemento
      ant = aux
      aux = LISTA[aux].Prox
    FimEnquanto
    atual = Livre
    Livre = LISTA[Livre].Prox
    LISTA[atual].Info = Elemento
    Se (ant = -1) então
      LISTA[atual].Prox = Cabec
      Cabec = atual
    senão
      LISTA[ant].Prox  =  atual
    fimse
    LISTA[atual].Prox = aux
  fimse
fimse

REMOÇÃO
Se Cabec = -1 então "REMOÇÃO IMPOSSÍVEL"
senão
  aux = Cabec
  ant = -1
  Enquanto aux != -1 e  LISTA[aux].Info != Elem
    ant = aux
    aux = LISTA[aux].Prox
  FimEnquanto
  Se aux = -1 então  "ERRO - Elem Inexistente"
  Senão
    Elemento = LISTA[aux].Info
    Se ant = -1 então
      Cabec = LISTA[aux].Prox
    senão
      LISTA[ant].Prox = LISTA[aux].Prox
    fimse
    LISTA[aux].Prox = Livre
    Livre = aux
  fimse
fimse
```

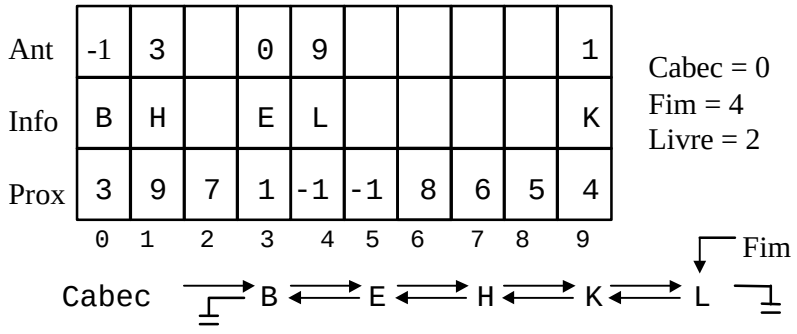
Exemplo 4.8 - Lista ordenada utilizando alocação dinâmica (ProjEx408.dpr e Ex408.pas)




```
Convenções:
Cabec = NULL indica lista vazia

INICIALIZAÇÃO
Cabec = NULL

INSERÇÃO
E = Alocar Espaço
E->Info = Elemento
```

```
E->Prox = NULL
aux = Cabec
ant = NULL
Enquanto aux != NULL e aux->Info < Elemento
  ant = aux
  aux = aux->Prox
FimEnquanto
Se ant = NULL então
  E->Prox = Cabec
  Cabec  = E
senão
  ant->Prox = E
  E->Prox = aux
FimSe

REMOÇÃO
Se Cabec = NULL então "LISTA VAZIA - REMOÇÃO IMPOSSÍVEL"
senão
  aux = Cabec
  ant = NULL
  Enquanto aux != NULL e aux->Info != Elemento
    ant = aux
    aux = aux->Prox
  FimEnquanto
  Se aux = NULL então "ELEM. INEXISTENTE - REMOÇÃO IMPOSSÍVEL"
  senão
    Elemento = aux->Info
    Se ant = NULL então
      Cabec = aux->Prox
    senão
      ant->Prox = aux->Prox
    fimse
    Liberar espaço (aux)
  fimse
fimse
```

Exemplo 4.9 - Lista Duplamente Ligada Ordenada utilizando estrutura com vetor (ProjEx409.dpr e Ex409.pas)

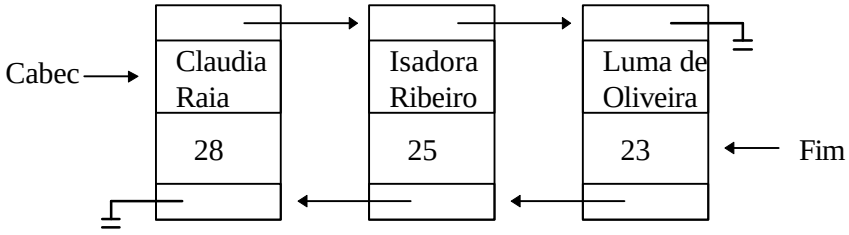



```
Convenções:
Cabec = -1 indica lista vazia
Livre = -1 indica lista cheia
TAM_VET =  10

INICIALIZAÇÃO
Cabec = -1
```

```
Fim   = -1
Livre =  0
Para  k = 0 até TAM_VET - 1 faça LDL[k].Prox = k+1
LDL[TAM_VET-1].Prox  =  -1

INSERÇÃO
Se Livre = -1 então "INSERÇÃO IMPOSSIVEL"
senão
  atual = Livre
  Livre = LDL[Livre].Prox
  LDL[atual].Info = Elemento
  LDL[atual].Prox = -1
  LDL[atual].Ant = -1
  Se Cabec = -1 então
    Cabec = Fim = atual
  senão
    aux = Cabec
    Enquanto aux != -1 e LDL[aux].Info < Elemento
      aux = LDL[aux].Prox
    FimEnquanto
    Se aux = -1 então
      LDL[atual].Ant = Fim
      LDL[Fim].Prox = atual
      Fim = atual
    senão
      LDL[LDL[aux].Ant].Prox  =  atual
      LDL[atual].Ant = LDL[aux].Ant
      LDL[atual].Prox = aux
      LDL[aux].Ant = atual
    fimse
  fimse
fimse

REMOÇÃO
Se Cabec = -1 então "REMOÇÃO IMPOSSÍVEL"
senão
  aux = Cabec
  Enquanto aux != -1 e LDL[aux].Info != Elem
    aux = LDL[aux].Prox
  FimEnquanto
  Se aux = -1 então  "ERRO - Elem Inexistente"
  senão
    Elemento = LDL[aux].Info
    Se aux = Cabec então
      Cabec = LDL[aux].Prox
      Se Cabec != -1 então
        LDL[Cabec].Ant = -1
      senão
        Cabec = Fim = -1
      fimse
    senão
    se aux = Fim
      LDL[LDL[aux].Ant].Prox = -1
      Fim = LDL[aux].Ant
    senão
      LDL[LDL[aux].Ant].Prox = LDL[aux].Prox
      LDL[LDL[aux].Prox].Ant = LDL[aux].Ant
    fimse
    LDL[aux].Prox = Livre
    Livre = aux
  fimse
```

```
fimse
```

Exemplo 4.10 - Lista Duplamente Ligada ordenada utilizando alocação dinâmica (ProjEx410.dpr e Ex410.pas)




```
INICIALIZAÇÃO
Cebec = NULL
Fim   = NULL

INSERÇÃO
E = Alocar Espaço
E->Info = Elemento
E->Prox = NULL
E->Ant  = NULL
Se Cebec = NULL então
  Cabec = Fim = E
senão
  aux = Cabec
  Enquanto  aux != NULL e aux->Info < Elemento
    aux = aux->Prox
  FimEnquanto
  Se aux = NULL então
    E->Ant = Fim
    Fim->Prox = E
    Fim = E
  senão
    E->Prox = aux
    Se aux = Cabec então
      aux->Ant = E
      Cabec  =  E
    senão
      aux->Ant->Prox = E
      E->Ant = aux->Ant
      E->Prox = aux
      aux->Ant = E
    fimse
  fimse
fimse

REMOÇÃO
Se  Cabec = NULL então "LISTA VAZIA - REM. IMPOSSÍVEL"
senão
  aux = Cabec
  Enquanto aux != 0 e aux->Info !=  Elemento
    aux = aux->Prox
  FimEnquanto
  Se aux = NULL então "ELEM NÃO EXISTE - REM. IMPOSSÍVEL"
  senão
    Elemento = aux->Info
```

```
    Se aux = Cabec
      Cabec = aux->Prox
      Se Cabec = NULL então
        Fim = NULL
      senão
        Cabec->Ant = NULL
      fimse
    senão
      se aux = Fim então
        Fim = Aux->Ant
        Fim->Prox = NULL
      senão
        aux->Ant->Prox = aux->Prox
        aux->Prox->Ant = aux->Ant
      fimse
    fimse
  fimse
  Elemento = aux->Info
  Liberar espaço (aux)
fimse
```

Exercícios

4.01) A estrutura abaixo contém uma lista ligada ordenada alfabeticamente: (exercício resolvido)

```
       00 01 02 03 04 05 06 07 08 09 10 11
Info    C --  K -- --  M --  B --  E --  F
Prox   09 -1 05 01 06 -1 10 00 03 11 08 02

 Cabec = 07   Fim = 05   Livre = 04
```

a) Inserir o elemento H

```
       00 01 02 03 04 05 06 07 08 09 10 11
Info    C --  K --  H  M --  B --  E --  F
Prox   09 -1 05 01 02 -1 10 00 03 11 08 04

 Cabec = 07   Fim = 05   Livre = 06
```

b) Inserir o elemento D

```
       00 01 02 03 04 05 06 07 08 09 10 11
Info    C --  K --  H  M  D  B --  E --  F
Prox   06 -1 05 01 02 -1 09 00 03 11 08 04

 Cabec = 07   Fim = 05   Livre = 10
```

c) Inserir o elemento A

```
       00 01 02 03 04 05 06 07 08 09 10 11
Info    C --  K --  H  M  D  B --  E  A  F
Prox   06 -1 05 01 02 -1 09 00 03 11 07 04

 Cabec = 10   Fim = 05   Livre = 08
```

d) Inserir o elemento P

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H  M  D  B  P  E  A  F
Prox  06 -1 05 01 02 08 09 00 -1 11 07 04

 Cabec = 10   Fim = 08   Livre = 03
```

e) Deletar o elemento M

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H --  D  B  P  E  A  F
Prox  06 -1 08 01 02 03 09 00 -1 11 07 04

 Cabec = 10   Fim = 08   Livre = 05
```

f) Deletar o elemento B

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H --  D --  P  E  A  F
Prox  06 -1 08 01 02 03 09 05 -1 11 00 04

 Cabec = 10   Fim = 08   Livre = 07
```

g) Deletar o elemento A

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H --  D --  P  E --  F
Prox  06 -1 08 01 02 03 09 05 -1 11 07 04

 Cabec = 00   Fim = 08   Livre = 10
```

h) Inserir o elemento N

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H --  D --  P  E  N  F
Prox  06 -1 10 01 02 03 09 05 -1 11 08 04

 Cabec = 00   Fim = 08   Livre = 07
```

i) Deletar o elemento P

```
      00 01 02 03 04 05 06 07 08 09 10 11
Info   C --  K --  H --  D -- --  E  N  F
Prox  06 -1 10 01 02 03 09 05 07 11 -1 04

 Cabec = 00   Fim = 10   Livre = 08
```

4.02) A estrutura abaixo contém 3 pilhas e 2 filas:
(exercício resolvido)

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  f -- --  a --  c --  B  A --  X --
Prox  03 08 13 00 10 14 09 00 00 02 16 01 00 07 04 06

 Topo1 = 05  Topo2= 15  Topo3 = 00  Livre= 11
 Cabec1= 12  Fim1 = 13  Cabec2= 00  Fim2 = 00
```

a) Inserir na pilha 3 o elemento z

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  f -- --  a --  c  z  B  A --  X --
Prox  03 08 13 00 10 14 09 00 00 02 00 01 00 07 04 06

 Topo1 = 05  Topo2= 15  Topo3 = 11  Livre= 16
 Cabec1= 12  Fim1 = 13  Cabec2= 00  Fim2 = 00
```

b) Remover da pilha 1

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y -- -- --  a -- -c  z  B  A --  X --
Prox  03 08 13 00 16 14 09 00 00 02 00 01 00 07 04 06

 Topo1 = 10  Topo2= 15  Topo3 = 11  LIvre= 05
 Cabec1= 12  Fim1 = 13  Cabec2= 00  Fim2 = 00
```

c) Inserir na fila 2 o elemento M

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M -- --  a --  c  z  B  A --  X --
Prox  03 08 13 00 00 14 09 00 00 02 00 01 00 07 04 06

 Topo1 = 10  Topo2= 15  Topo3 = 11  Livre= 16
 Cabec1= 12  Fim1 = 13  Cabec2= 05  Fim2 = 05
```

d) Inserir na fila 1 o elemento D

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M -- --  a --  c  z  B  A --  X  D
Prox  03 08 13 00 00 14 09 00 00 02 00 01 16 07 04 00

 Topo1 = 10  Topo2= 15  Topo3 = 11  Livre= 06
 Cabec1= 12  Fim1 = 16  Cabec2= 05  Fim2 = 05
```

e) Inserir na fila 2 o elemento N

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N --  a --  c  z  B  A --  X  D
Prox  03 08 13 00 06 00 09 00 00 02 00 01 16 07 04 00

 Topo1 = 10  Topo2= 15  Topo3 = 11  Livre= 14
 Cabec1= 12  Fim1 = 16  Cabec2= 05  Fim2 = 06
```

f) Deletar da fila 1

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N --  a --  c  z --  A --  X  D
Prox  03 08 13 00 06 00 09 00 00 02 00 14 16 07 04 00

 Topo1 = 10  Topo2= 15  Topo3 = 11  Livre= 12
 Cabec1= 01  Fim1 = 16  Cabec2= 05  Fim2 = 06
```

g) Inserir na pilha 2 o elemento W

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N --  a --  c  z  W  A --  X  D
Prox  03 08 13 00 06 00 09 00 00 02 00 15 16 07 04 00

 Topo1 = 10  Topo2= 12  Topo3 = 11  Livre= 14
 Cabec1= 01  Fim1 = 16  Cabec2= 05  Fim2 = 06
```

h) Inserir na fila 2 o elemento P

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N --  a --  c  z  W  A  P  X  D
Prox  03 08 13 00 06 14 09 00 00 02 00 15 16 00 04 00

 Topo1 = 10  Topo2= 12  Topo3 = 11  Livre= 07
 Cabec1= 01  Fim1 = 16  Cabec2= 05  Fim2 = 14
```

i) Inserir na pilha 1 o elemento h

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N  h  a --  c  z  W  A  P  X  D
Prox  03 08 13 00 06 14 10 00 00 02 00 15 16 00 04 00

 Topo1 = 07  Topo2= 12  Topo3 = 11  Livre= 09
 Cabec1= 01  Fim1 = 16  Cabec2= 05  Fim2 = 14
```

j) Inserir na fila 1 o elemento G

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Info   F  e  C  Y  M  N  h  a  G  c  z  W  A  P  X  D
Prox  03 08 13 00 06 14 10 00 00 02 00 15 16 00 04 09

 Topo1 = 07  Topo2= 12  Topo3 = 11  Livre= 00
 Cabec1= 01  Fim1 = 09  Cabec2= 05  Fim2 = 14
```

4.03) A estrutura abaixo contém uma lista duplamnete ligada ordenada alfabeticamente (FrenteDL e FimDL) e uma fila circular (FimCirc):

Obs1: note que a fila circular necessita apenas de um apontador para o fim da mesma.

Obs2: a fila circular não utiliza o campo anterior.

(exercício resolvido)

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 10 -- 03 -- -- -- -- 00 -- 15 -- -- 05 --
Info   z  R  F  x  H -- --  a --  C --  N  w --  K --
Prox  13 00 05 08 15 14 00 01 07 03 09 02 04 16 12 11
-------------------------------------------------------
 FrenteDL = 10  FimDL = 02  FimCirc = 13  Livre = 06
```

a) Inserir na lista dup. ligada o elemento D

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- 03 10 -- -- -- 00 -- 15 -- -- 05 --
Info   z  R  F  x  H  D --  a --  C --  N  w --  K --
Prox  13 00 05 08 15 03 00 01 07 06 09 02 04 16 12 11
-------------------------------------------------------
 FrenteDL = 10  FimDL = 02  FimCirc = 13  Livre = 14
```

b) Inserir na lista dup. ligada o elemento M

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- 03 10 -- -- -- 00 -- 14 -- 15 05 --
Info   z  R  F  x  H  D --  a --  C --  N  w  M  K --
Prox  13 00 05 08 15 03 00 01 07 06 09 02 04 12 14 11
-------------------------------------------------------
 FrenteDL = 10  FimDL = 02  FimCirc = 13  Livre = 16
```

c) Inserir na fila circular o elemento b

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- 03 10 -- -- -- 00 -- 14 -- 15 05 --
Info   z  R  F  x  H  D --  a --  C --  N  w  M  K  b
Prox  13 00 05 08 15 03 00 01 07 06 09 02 16 12 14 04
-------------------------------------------------------
 FrenteDL = 10  FimDL = 02  FimCirc = 16  Livre = 11
```

d) Inserir na fila circular o elemento y

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- 03 10 -- -- -- 00 -- 14 -- 15 05 --
Info   z  R  F  x  H  D --  a --  C  y  N  w  M  K  b
Prox  13 00 05 08 15 03 00 01 07 06 04 02 16 12 14 11
-------------------------------------------------------
 FrenteDL = 10  FimDL = 02  FimCirc = 11  Livre = 09
```

e) Inserir na lista dup. ligada o elemento A

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- 03 10 -- -- 00 09 -- 14 -- 15 05 --
Info   z  R  F  x  H  D --  a  A  C  y  N  w  M  K  b
Prox  13 00 05 08 15 03 00 01 10 06 04 02 16 12 14 11
-------------------------------------------------------
```

```
 FrenteDL = 09  FimDL = 02  FimCirc = 11  Livre = 07
```

f) Remover da lista dup. ligada o elemento H

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- -- 10 -- -- 00 09 -- 14 -- 15 03 --
Info   z  R  F  x --  D --  a  A  C  y  N  w  M  K  b
Prox  13 00 15 08 07 03 00 01 10 06 04 02 16 12 14 11
-------------------------------------------------------
 FrenteDL = 09  FimDL = 02  FimCirc = 11  Livre = 05
```

g) Remover da fila circular

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- -- 10 -- -- 00 09 -- 14 -- 15 03 --
Info   z  R  F -- --  D --  a  A  C  y  N  w  M  K  b
Prox  13 00 15 05 07 03 00 01 10 06 08 02 16 12 14 11
-------------------------------------------------------
 FrenteDL = 09  FimDL = 02  FimCirc = 11  Livre = 04
```

h) Remover da fila circular

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- -- 10 -- -- 00 09 -- 14 -- 15 03 --
Info   z  R  F -- --  D -- --  A  C  y  N  w  M  K  b
Prox  13 00 15 05 07 03 00 04 10 06 01 02 16 12 14 11
-------------------------------------------------------
 FrenteDL = 09  FimDL = 02  FimCirc = 11  Livre = 08
```

i) Remover da lista dup. ligada o elemento K

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- -- 10 -- -- 00 09 -- 14 -- 03 -- --
Info   z  R  F -- --  D -- --  A  C  y  N  w  M --  b
Prox  13 00 14 05 07 03 00 04 10 06 01 02 16 12 08 11
-------------------------------------------------------
 FrenteDL = 09  FimDL = 02  FimCirc = 11  Livre = 15
```

j) Insira na lista dup. ligada o elemento T

```
      01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
-------------------------------------------------------
Ant   -- 12 06 -- -- 10 -- -- 00 09 -- 14 -- 03 02 --
Info   z  R  F -- --  D -- --  A  C  y  N  w  M  T  b
Prox  13 15 14 05 07 03 00 04 10 06 01 02 16 12 00 11
-------------------------------------------------------
 FrenteDL = 09  FimDL = 15  FimCirc = 11  Livre = 08
```

4.04) A estrutura abaixo contém uma LISTA ordenada alfabeticamente

```
#Prox  Info
01     06    -
02     00    K          Cabec = 03
03     08    C
04     01    -          Fim = 02
05     02    H
06     07    -          Livre = 04
07     09    -
08     05    F
09     10    -
10     00    -
```

Efetue as seguintes operações na estrutura redesenhando a mesma a cada passo
a) Insira o elemento D
b) Insira o elemento M
c) Insira o elemento B
d) Remova o elemento F
e) Insira o elemento A
f) Remova o elemento M

4.05) A estrutura abaixo contém duas pilhas e duas filas. As pilhas são referenciadas pelas variáveis **Topo1** e **Topo2** enquanto as filas são acessadas através das variáveis **Inic1**, **Fim1**, **Inic2** e **Fim2**.

```
#Prox  Info
01     13    F
02     15    a          Topo1 = 02
03     01    D
04     00    z          Topo2 = 10
05     00    -
06     17    M          Inic1 = 03
07     11    c          Fim1  = 17
08     05    -
09     14    Y          Inic2 = 09
10     19    x          Fim2  = 14
11     00    b
12     08    -          Livre = 18
13     06    A
14     00    R
15     07    d
16     12    -
17     00    B
18     20    -
19     04    k
20     16    -
```

Efetue as seguintes operações na estrutura redesenhando a mesma a cada passo
a) Insira o elemento < **f** > na pilha 01
b) Insira o elemento < **w** > na pilha 02
c) Insira o elemento < **H** > na fila 02
d) Insira o elemento < **J** > na fila 01
e) Remova da fila 01
f) Remova da fila 02
g) Insira o elemento < **n** > na pilha 01
h) Remova da pilha 02

4.06) A estrutura abaixo manipula uma PILHA, uma FILA, uma LISTA CIRCULAR ordenada alfabeticamente e uma LISTA DUPLAMENTE LIGADA também ordenada alfabeticamente.

```
#     Prox  Info  Ant
01    15    N     17              Livre = 18
02    06    F     --
03    05    e     --                          1   2   3   4
04    00    J     --              Cabeça = [ 03  11  10  17 ]
05    09    b     --
06    04    H     --
07    25    --    --                          1   2   3   4
08    16    o     --              Fim    = [ 21  04  16  20 ]
09    21    f     --
10    14    m     --
11    24    B     --              Cabeça[1] = 03 e Fim[1] = 21 são
12    22    --    --              respectivamente o início e o fim
13    20    R     15              de uma PILHA
14    08    n     --
15    13    P     01              Cabeça[2] = 11 e Fim[2] = 04 são
16    10    q     --              respectivamente o início e o fim
17    01    M     00              de uma FILA
18    30    --    --
19    12    --    --              Cabeça[3] = 10 e Fim[3] = 16 são
20    00    S     13              respectivamente o início e o fim
21    00    g     --              de um LISTA CIRCULAR ordenada
22    23    --    --              alfabeticamente
23    07    --    --
24    02    C     --              Cabeça[4] = 17 e Fim[4] = 20 são
25    27    --    --              respectivamente o início e o fim
26    00    --    --              de um LISTA DUPLAMENTE LIGADA
27    26    --    --              ordenada alfabeticamente
28    19    --    --
29    28    --    --
30    29    --    --
```

a) Efetue as operações abaixo na ordem dada e modifique a estrutura acima segundo as alterações realizadas

- inserir < a > na PILHA
- inserir < K > na FILA
- inserir < l > e < t > na LISTA CIRCULAR
- inserir < Q > e < T > na LISTA DUPLAMENTE LIGADA

obs: redesenhe a estrutura acima de modo a refletir a inserções acima

b) Efetue as operações abaixo na ordem dada e modifique a estrutura acima segundo as alterações realizadas

- deletar < B > da FILA
- deletar < m > e < q > da LISTA CIRCULAR
- deletar < R > da LISTA DUPLAMENTE LIGADA

obs: redesenhe a estrutura acima de modo a refletir a remoções acima

4.07) Alterar os algoritmos de inicialização, inserção e remoção para lista linear (ordenada alfabeticamente) dados em aula de modo a implementar uma lista linear

circular (também ordenada). Alterar tanto os algoritmos que trabalham com arranjo como os que trabalham com alocação dinâmica.

4.08) Utilizando a estrutura de dados baseada em apontadores utilizando alocação dinâmica, escreva algoritmos que realizem as seguintes operações:

- Concatenar duas listas (juntar uma no final da outra)
- Remover todos os elementos de uma lista
- Inverter uma lista (pode-se utilizar recursividade)
- Combinar duas listas ordenadas numa única lista ordenada
- Acessar o n-ésimo elemento de uma lista (busca do n-ésimo elemento)
- Acessar um elemento particular da lista (busca de um elemento qualquer)
- Contar o número de elementos de uma lista
- Fazer uma cópia da lista
- Montar uma nova lista contendo a intersecção de duas outras listas
- Montar uma nova lista contendo a união de duas outras listas
- Inserir um elemento depois do n-ésimo elemento da lista
- Deletar o n-ésimo elemento de uma lista
- Ordenar uma lista (segundo algum critério)

Obs 1: A lista possui um apontador para o início da mesma chamado **Cabec** e um apontador para o último elemento chamado **Fim**.

Obs 2: Caso você tenha que trabalhar com mais de uma lista, crie mais alguns nomes para especificar os inícios e finais das mesmas (ex: **Cabec1**, **Fim1**, **Cabec2**, **Fim2**, etc)

4.09) Utilizando a estrutura de dados baseada em apontadores utilizando um vetor previamente alocado, reescreva os algoritmos do exercício anterior.

Obs 1: A lista possui uma variável indicando o início da mesma (chamada **Cabec**) e uma outra variável referenciando o último elemento (chamada **Fim**). Também existe uma variável chamada **Livre** indicando os espaços livres (como dado em aula).

Obs 2: Caso você tenha que trabalhar com mais de uma lista, crie mais alguns nomes para especificar os inícios e finais das mesmas (ex: **Cabec1**, **Fim1**, **Cabec2**, **Fim2**, etc)