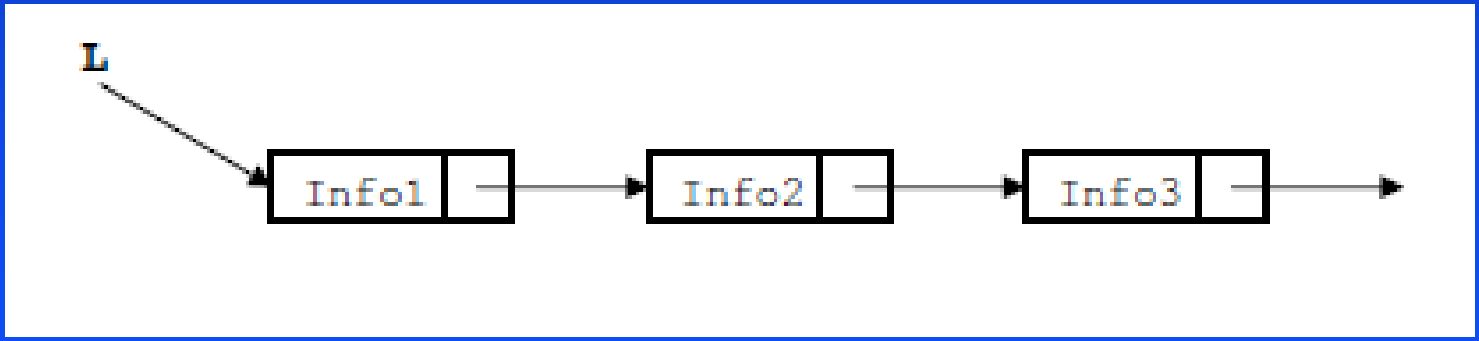
Estruturas de Dados I

Listas Encadeadas
ou
Listas Ligadas

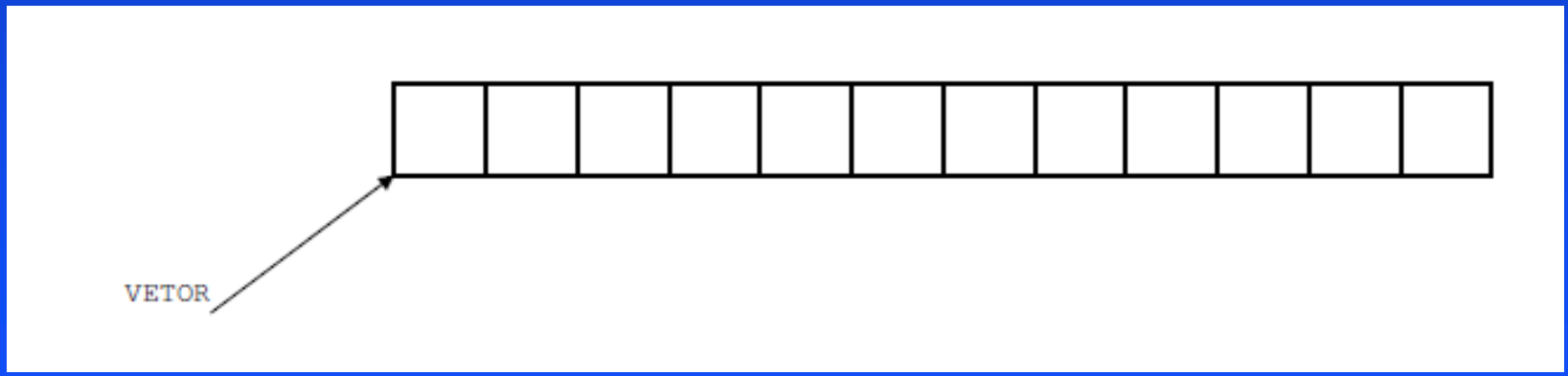
Listas encadeadas ou lista ligada

- Uma **lista encadeada,** ou **lista ligada,** é uma estrutura de dados linear e dinâmica. Ela é composta por células que **apontam** para o próximo elemento da lista. Para "ter" uma lista encadeada, basta guardar seu **primeiro elemento**, e seu último elemento aponta para uma célula nula. O esquema a seguir representa uma lista ligada/encadeada com 3 elementos:



Porque utilizar listas encadeadas e não vetores?

- Vetor
  - ocupa um espaço contíguo de memória
  - permite acesso randômico aos elementos
  - deve ser dimensionado com um número máximo de elementos



Listas encadeadas

- Lista encadeada:
  - sequência encadeada de elementos, chamados de *nós da lista*
  - nó da lista é representado por dois campos:
- a informação armazenada e o ponteiro para o próximo elemento da lista
  - a lista é representada por um ponteiro para o primeiro nó e;
  - o ponteiro do último elemento é NULL

Listas encadeadas

- Exemplo:
  - lista encadeada armazenando valores inteiros
  - – estrutura lista
    - estrutura dos nós da lista
  - – tipo Lista
    - tipo dos nós da lista

```
struct lista {
   int info;
   struct lista* prox;
};
typedef struct lista Lista;
```

lista é uma estrutura auto-referenciada, pois o campo *prox* é um ponteiro para uma próxima estrutura do mesmo tipo

uma lista encadeada é representada pelo ponteiro para seu primeiro elemento, do tipo Lista*

Listas encadeadas

- Pode ser feita a declaração de uma vez só.

- Por exemplo:

```
typedef struct lista {
int info;
lista *prox;
} Lista;
```

Listas encadeadas

- Exemplo - Função de criação
  - cria uma lista vazia, representada pelo ponteiro NULL

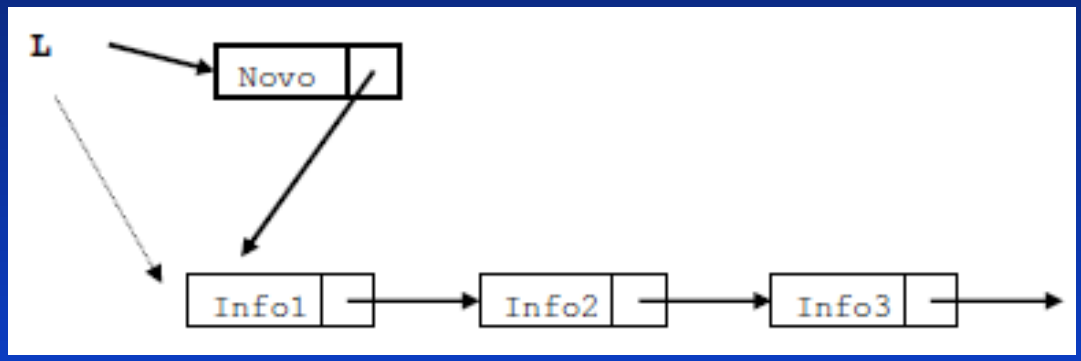
```
/* função de criação: retorna uma lista vazia */
Lista* criaLista ()
{
return NULL;
}
```

Listas encadeadas

- Exemplo - Função de inserção

– aloca memória para armazenar o elemento

– encadeia o elemento na lista existente




```
/* inserção no início: retorna a lista atualizada */
Lista *insereLista (Lista *l, int i)
{
Lista *novo = (Lista*) malloc(sizeof(Lista));
novo->info = i;
novo->prox = l;
return novo;
}
```

Listas encadeadas

- Exemplo:
  - Cria uma lista inicialmente vazia e insere novos elementos:

```
int main() {
    Lista *aLista=criaLista(); //cria a lista vazia
    aLista = insereLista(aLista,23); //insere 23
    aLista = insereLista(aLista,45); //insere 45
    // ... Adiciona todos os elementos
   return 0;
}
```

Listas encadeadas

- Exemplo:
  - Mostrar todos os elementos da lista:

```
// função imprime: imprime valores dos elementos
void imprimeLista (Lista *l)
{
    Lista *p;
    for (p = l; p != NULL; p = p->prox)
        printf("info = %d\n", p->info);
}
```

Listas encadeadas

- Exemplo:
  - Verificar se a lista está vazia:

```
/* função vazia: retorna 1 se vazia ou 0 se não vazia */
int ListaVazia (Lista *l)
{
   return (l == NULL);
}
```

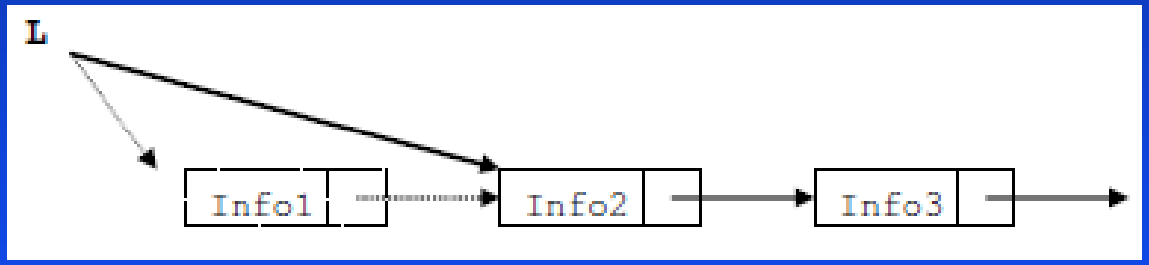
Listas encadeadas

- Exemplo:
  - Buscar um elemento na lista:

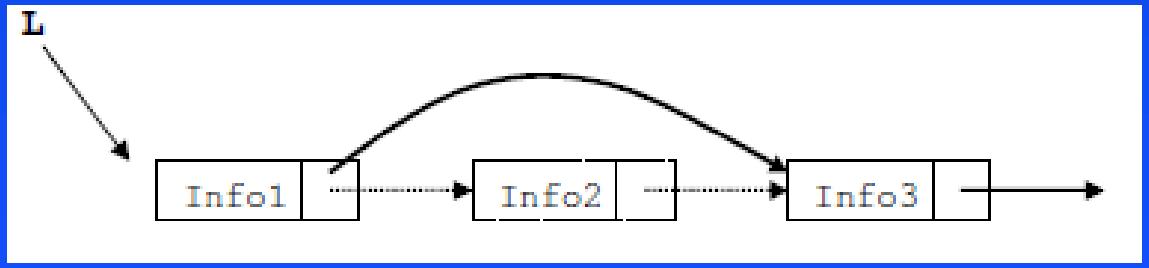
```
/* função busca: busca um elemento na lista */
Lista *busca (Lista *l, int v)
{
Lista *p;
for (p=l; p!=NULL; p = p->prox)
{
   if (p->info == v)
      return p;
}
return NULL; /* não achou o elemento */
}
```

Listas encadeadas

- Exemplo:
  - Remover elemento da lista.
- recebe como entrada a lista e o valor do elemento a retirar
- atualiza o valor da lista, se o elemento removido for o primeiro



- caso contrário, apenas remove o elemento da lista



Listas encadeadas

```
/* função retira: retira elemento da lista */
Lista *Remove (Lista *l, int v)
{
  Lista *ant = NULL; /* ponteiro para elemento anterior */
  Lista *p = l; /* ponteiro para percorrer a lista */
  /* procura elemento na lista, guardando anterior */
  while (p != NULL && p->info != v)
  {
    ant = p;
    p = p->prox;
  }
  /* verifica se achou elemento */
  if (p == NULL)
  {
     printf("\nElemento nao encontrado\n");
     return l; /* não achou: retorna lista original */
  }
  //continua...
```

```
//continuação

/* retira elemento */
  if (ant == NULL)
  { /* retira elemento do inicio */
    l = p->prox;
    printf("\nElemento removido \n");
  }
  else
  {
    /* retira elemento do meio da lista
*/
    ant->prox = p->prox;
    printf("\nElemento removido \n");
  }
  free(p);
  return l;
}
```

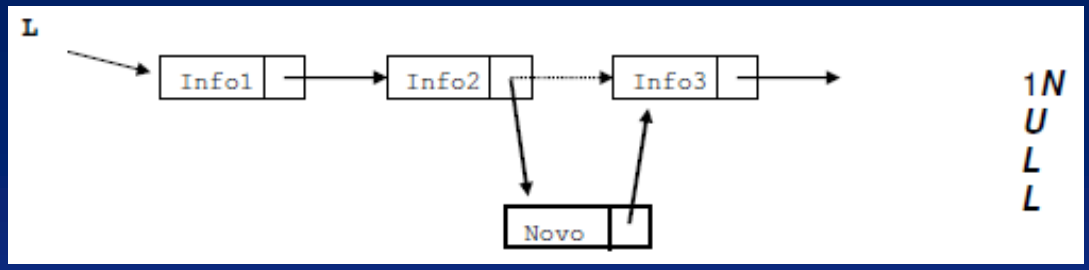
Listas encadeadas

- Exemplo:
  - Função libera lista (liberar a memória para ser reutilizada).

- Destrói a lista, liberando todos os elementos alocados

```
void liberaLista (Lista *l)
{
Lista *p = l, *t;
while (p != NULL)
   {
      t = p->prox; /* guarda referência p/ próx. elemento */
      free(p); /* libera a memória apontada por p */
      p = t; /* faz p apontar para o próximo */
   }
}
```

Listas encadeadas



- Exemplo:
  - Função insere_ordenado. Pode ser facilmente adaptada para inserir elementos no meio da lista, basta definir o critério.

```
Lista *insere_ordenado (Lista *l, int v)
{
   Lista *novo;
   Lista *ant = NULL; // ptr p/ elemento anterior
   Lista *p = l; // ponteiro para percorrer a lista
   /* procura posição de inserção */
   while (p != NULL && p->info < v)
   {
       ant = p;
       p = p->prox;
   }
/* cria novo elemento */
   novo = (Lista*) malloc(sizeof(Lista));
//cotinua ....
```

```
//...continuação
novo->info = v;
   /* encadeia elemento */
   if (ant == NULL)
   { /* se o anterior é vazio, insere
elemento no início */
      novo->prox = l; l = novo;
   }
   else
   { // insere elemento no meio da lista
      novo->prox = ant->prox;
      ant->prox = novo;
   }
   return l;
}
```

Listas encadeadas

- Exercícios:

1. Implementar as funções aqui apresentadas e incluir um Menu para o usuário poder acionar as opções como ele desejar.

2. Crie uma função para inserir um elemento no final da lista encadeada.

Listas encadeadas

3. Crie uma função para remover o último elemento da lista.

==========

Para estudos complementares: agenda que guarde os nomes dos contatos (em ordem alfabética), telefone e email programa de listas encadeadas da aula.