Anhanguera

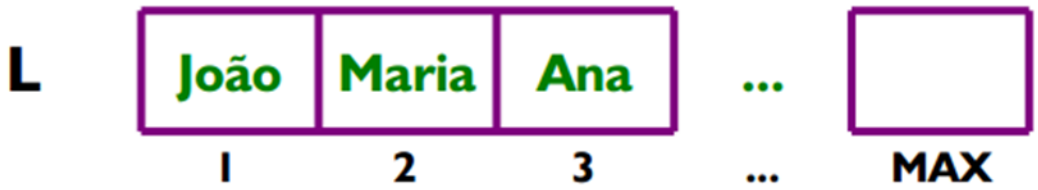
Aqui o seu esforço ganha força.

Listas Simplesmente Encadeadas

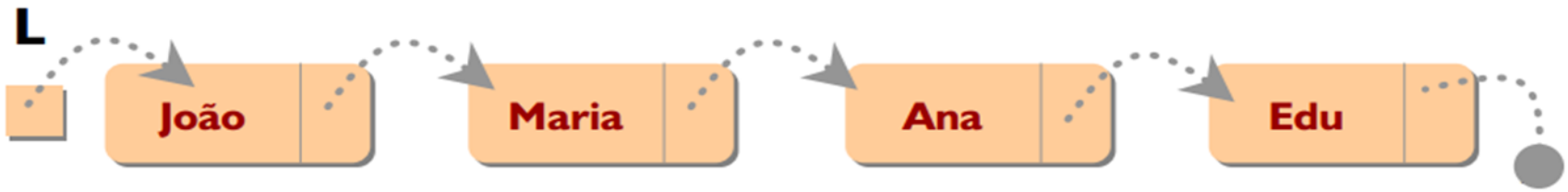
Prof. Esp. Rodrigo Hentz

Tipos de listas

- Veremos os conceitos de aplicação de dois tipos de listas:
- Sequencial e Encadeada
- Na lista sequencial os nós são armazenados em endereços sequencias, como um vetor.



- Na lista encadeada os nós são sequenciados através de ponteiros.



Listas Simplesmente Encadeadas

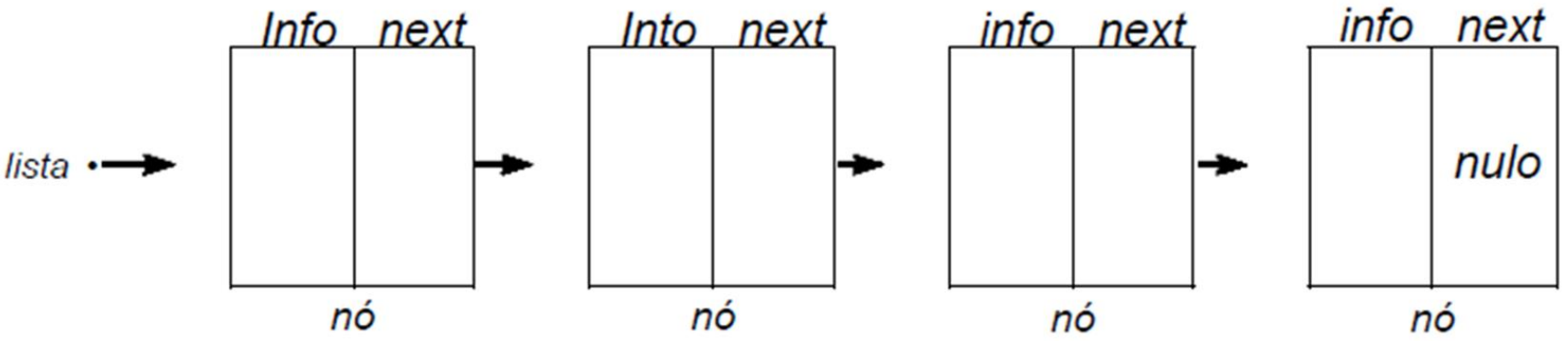
- Também encontradas na literatura com o nome de listas ligadas.
- Em uma lista simplesmente encadeada, para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo.
- Fazemos uso da alocação dinâmica de memória.
- Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenado.

Listas Simplesmente Encadeadas

- No entanto, não podemos garantir que os elementos armazenados na lista ocuparão um espaço de memória contíguo, portanto não temos acesso direto aos elementos da lista.
- Para que seja possível percorrer todos os elementos da lista, devemos explicitamente guardar o encadeamento dos elementos, o que é feito armazenando-se, junto com a informação de cada elemento, um ponteiro para o próximo elemento da lista.

Listas Simplesmente Encadeadas

- Cada item na lista é chamada de nó e contém dois campos, um campo de informação e um campo do endereço do próximo nó.

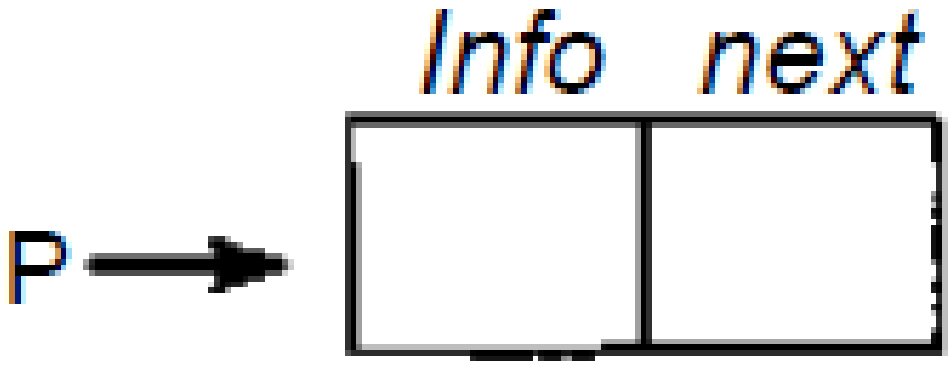


Anhanguera

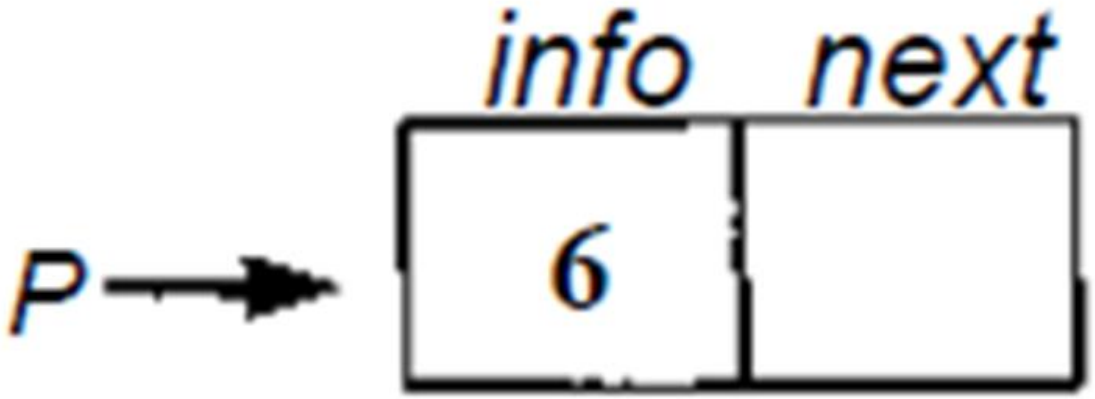
Operações

- Incluir nó (inicio ou fim).
- Excluir nó.
- Incluir nó ordenado.
- Pesquisar nó.
- Imprimir nós.

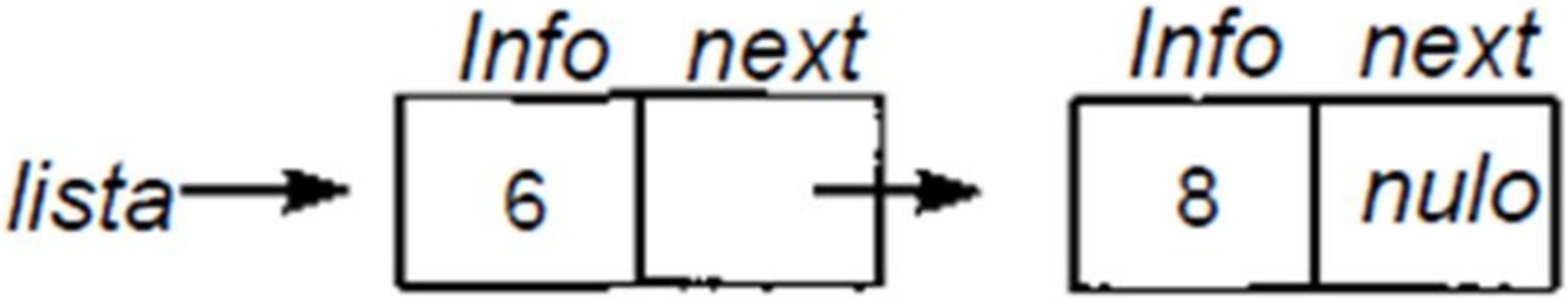
Nó



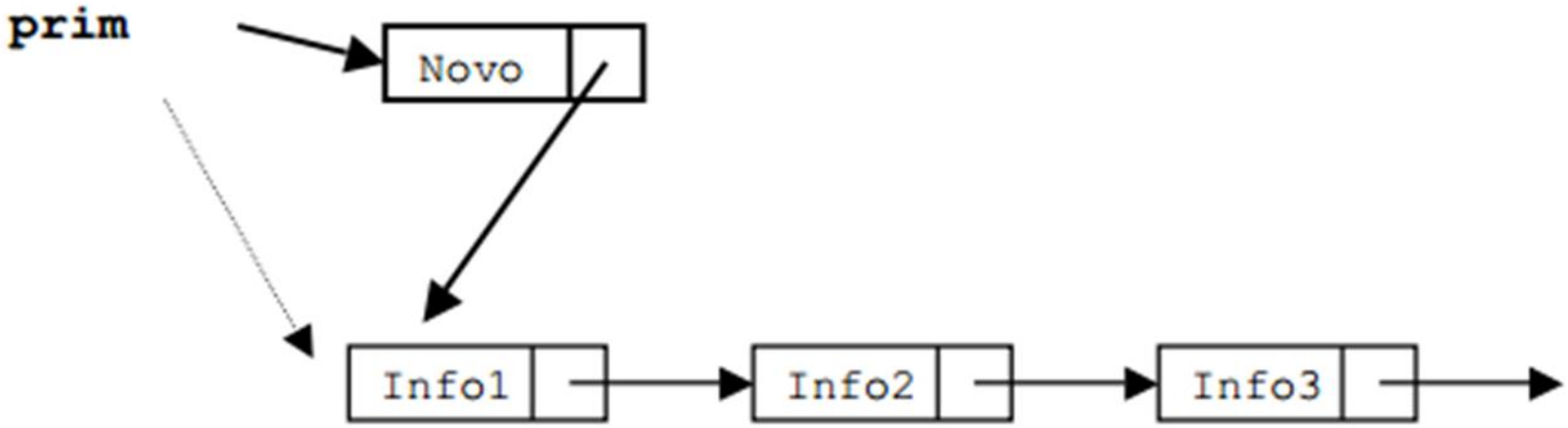
Nó



Nó

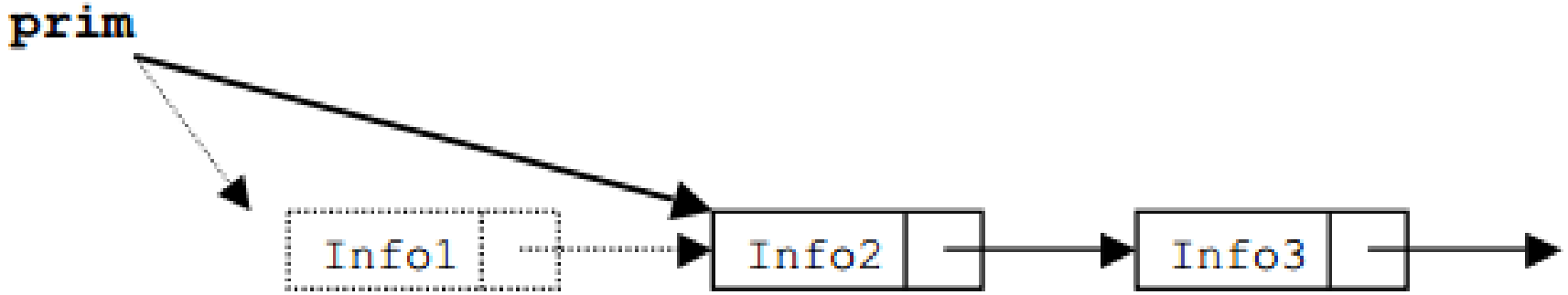


Incluir nó no início



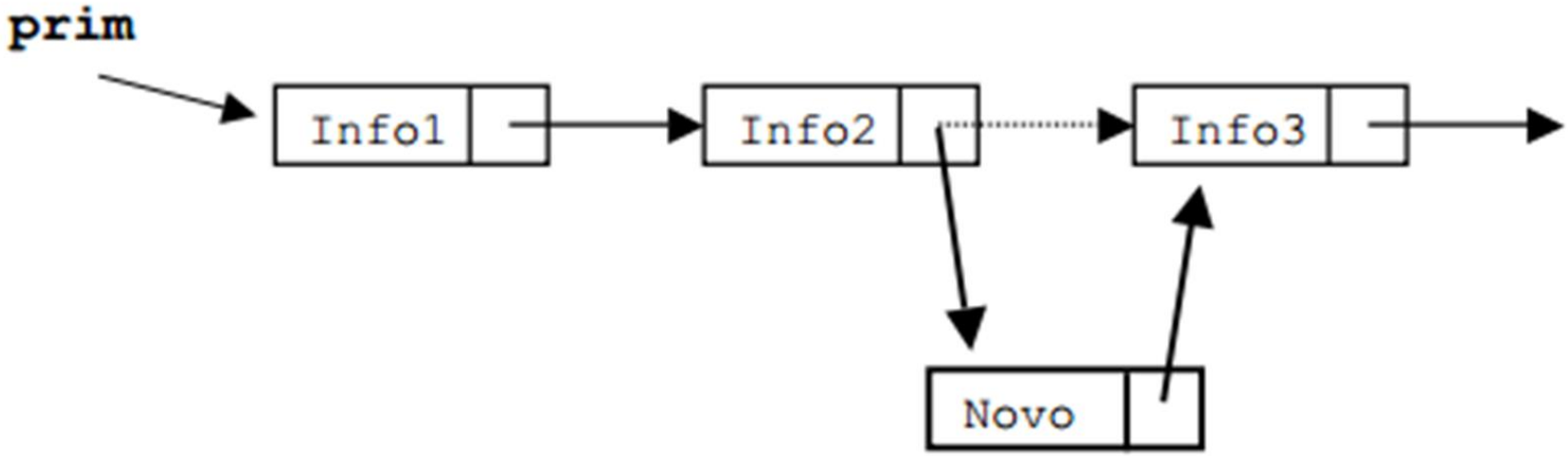
Excluir nó

A)



B)

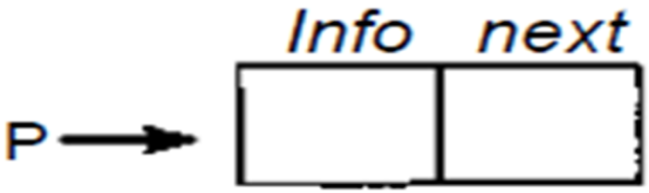
Incluir nó ordenado



Anhanguera

Criação da estrutura nó

```
typedef struct no {
   int info;
   struct no* next;
} sLista, sNo;
```



Anhanguera

Iniciar a lista

```
sLista* inicializaLista()
{
    printf("\nLista criada.");
    return NULL;
}
```

Iniciar a lista menu

```
int main(int argc, char *argv[]) {

    int opcao;
    sLista* lista;
    do
```

```
case 1:
    lista = inicializaLista();
    break;
case 2:
    break;
```

Anhanguera

Criar nó

```
sNo* criarNo(int valor)
{
   sNo* p = (sNo*)malloc(sizeof(sNo));
   p->info = valor;
   p->next = NULL;
   return p;
}
```

Anhanguera

Inserir no início

```
sLista* inserirInicio(sNo* no, int valor)
{
   sNo* novo = criarNo(valor);
   novo->next = no;
   return novo;
}
```

Anhanguera

Inserir no início menu

```
int main(int argc, char *argv[]) {

    int opcao, num;
```

```
case 2:
    printf ("\nEntre com o numero: ");
    scanf ("%d", &num);
    lista = inserirInicio(lista, num);
    break;
```

Anhanguera

Imprimir lista

```
void imprimirLista(sLista* lista)
{
   sNo* p;
   for (p = lista; p != NULL; p = p->next)
      printf("info = %d\n", p->info);
}
```

Menu

```
case 4:
      imprimirLista(lista);
      break;
```

Anhanguera

Excluir nó

```
sLista* excluirNo(sLista* lista, int valor)
{
  sNo* ant = NULL; /* ponteiro para elemento anterior */
  sNo* p = lista;  /* ponteiro para percorrer a lista*/
  while (p != NULL && p->info != valor) {
    /* procura elemento na lista, guardando anterior */
    ant = p;
    p = p->next;
  }
  /* verifica se achou elemento */
  if (p == NULL) /* não achou: retorna lista original */
    return lista;
  if (ant == NULL) {
    /* retira elemento do inicio */
   lista = p->next;
  }
  else
  {
    /* retira elemento do meio da lista */
    ant->next = p->next;
  }
  free(p);
  return lista;
}
```

Anhanguera

Excluir nó menu

```
case 3:
    printf ("\nEntre com o numero para remover: ");
    scanf ("%d", &num);
    lista = excluirNo(lista, num);
    break;
```

Buscar nó

```
sNo* buscarNo(sLista* lista, int valor)
{
   sNo* p;
   for (p=lista; p!=NULL; p=p->next)
   if (p->info == valor) return p;
   return NULL;
}
```

Anhanguera

Buscar nó menu

```
int main(int argc, char *argv[]) {

    int opcao, num;
    sLista* lista;
    sNo* no;
    do

case 5:
    printf ("\nEntre com o numero para pesquisa: ");
    scanf ("%d", &num);
    no = buscarNo(lista, num);
    if (no == NULL) printf("\nValor não encontrado.");
    else printf("\nValor encontrado.");
    break;
```

Anhanguera

Insere ordenado

```
sLista* insereOrdenado(sLista* lista, int valor)
{
   sNo* novo = criarNo(valor); /* cria novo nó */
   sNo* ant = NULL; /* ponteiro para elemento anterior */
   sNo* p = lista; /* ponteiro para percorrer a lista*/
   /* procura posição de inserção */
   while (p != NULL && p->info < valor) {
      ant = p;
      p = p->next;
   }
   /* insere elemento */
   if (ant == NULL) { /* insere elemento no início */
      novo->next = lista;
      lista = novo;
   }
   else { /* insere elemento no meio da lista */
      novo->next = ant->next;
      ant->next = novo;
   }
   return lista;
}
```

Anhanguera

Insere ordenado menu

```
case 7:
    printf ("\nEntre com o numero para o novo no: ");
    scanf ("%d", &num);
    lista = insereOrdenado(lista, num);
    break;
```

Liberar a lista

```
sLista* liberarLista(sLista* lista)
{
  sNo* p = lista;
  while (p != NULL) {
    sNo* t = p->next; /* guarda referência para o próximo elemento */
    free(p); /* libera a memória apontada por p */
    p = t; /* faz p apontar para o próximo */
  }
  return p;
}
```

Menu

```
case 6:
      lista = liberarLista(lista);
      printf("\nLista liberada.");
      break;
```

Anhanguera

Exercício

- No programa de lista linear encadeada implementar uma função que altere o valor de determinada posição.

  Exemplo:

  Na lista temos os valores: 20 | 25 | 30

  Quero alterar o valor do nó 25 para 40.

  Resultado: 20 | 40 | 30

- Criar uma função para inserir um valor no final da lista
- Criar uma função para exibir o tamanho total da lista.

Finalização