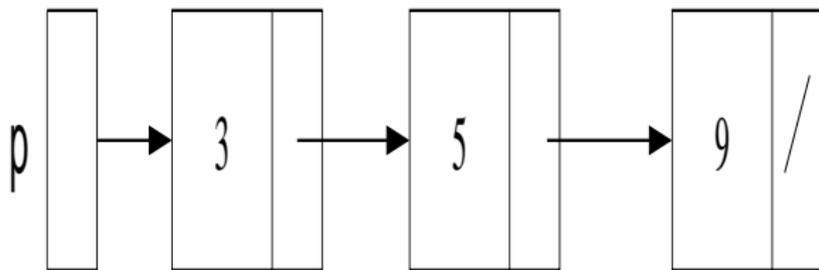


MC-102 — Aula 19

Listas Ligadas I

Instituto de Computação – Unicamp

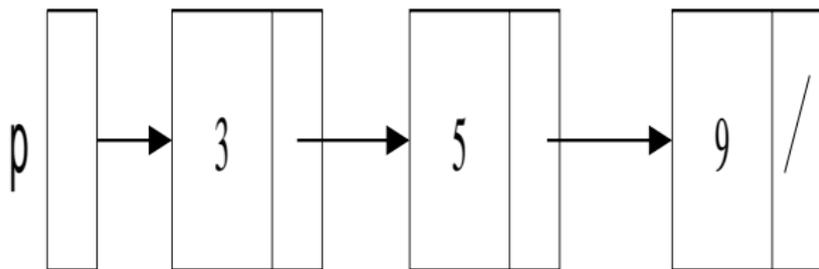
Segundo Semestre de 2009



Sequência de nós, em que cada nó possui

- um ou mais campos de informação e
- um apontador para o próximo nó da lista.

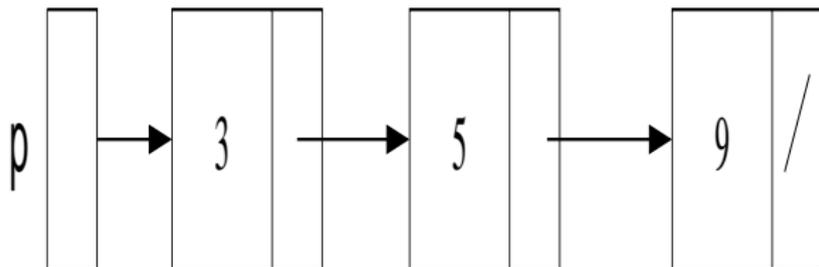
Estrutura de Lista Ligada



```
struct no {  
    int v;  
    struct no *prox;  
};  
typedef struct no No;  
  
No c; //c.v e c.prox  
No *p; //p->v e p->prox
```

Veja o código: lista.c

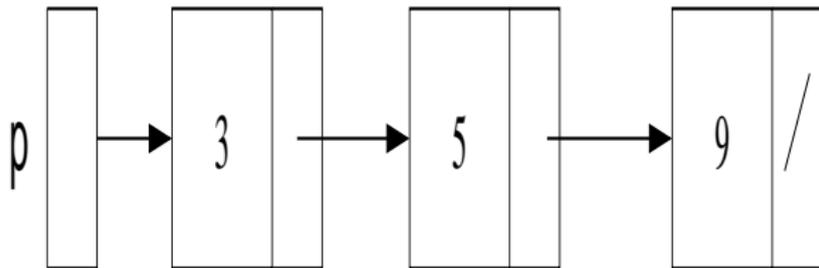
Endereço de uma lista ligada



- O endereço de uma lista é o endereço de seu primeiro nó
- A lista é vazia se o endereço de seu primeiro nó é NULL
- Para criar uma lista vazia:

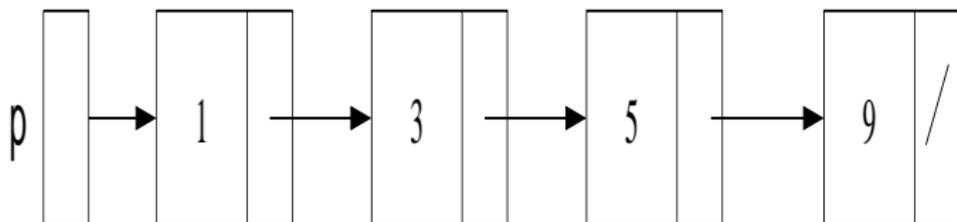
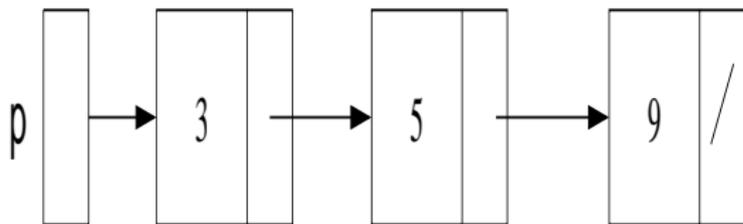
```
No lista;  
lista = NULL;
```

Imprimindo uma lista



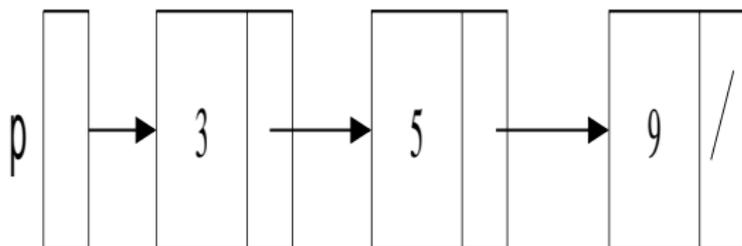
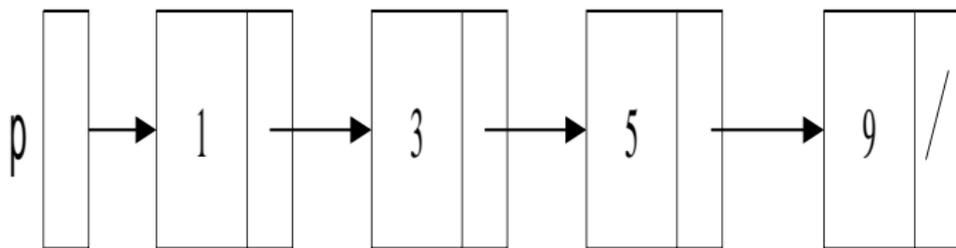
```
void imprime_lista(No *lista){
    No *p=lista;
    while (p!=NULL){
        printf("%d \n", p->v);
        p = p->prox;
    }
}
```

Inserção no começo da lista



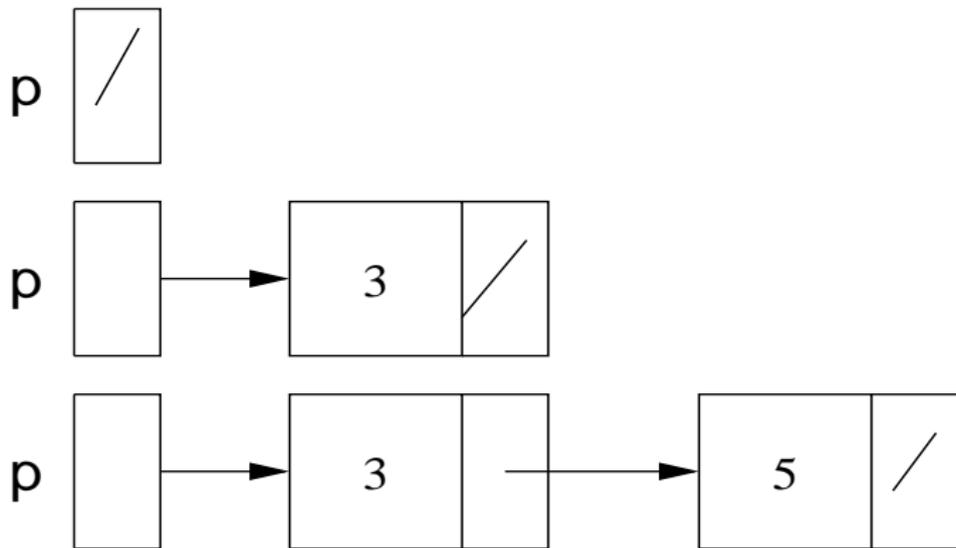
- o endereço armazenado em `p` deve ser alterado
- Veja o código: `insere_comeco.c`

Remoção no começo da lista



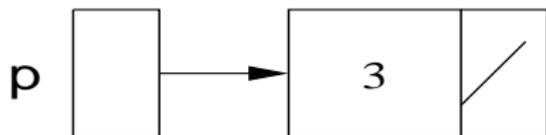
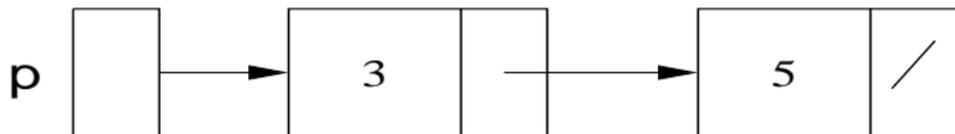
- o endereço armazenado em `p` deve ser alterado
- Veja o código: `remove_comeco.c`

Inserção no final da lista



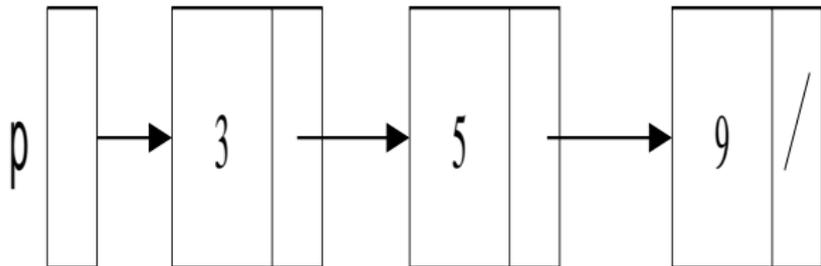
- o endereço armazenado em `p` será alterado caso a lista esteja vazia ou
- o campo `prox` do último elemento será alterado.
- Veja o código: `insere_final.c`

Remoção no final da lista



- o campo prox do último elemento será alterado caso a lista contenha mais de um elemento ou
- o endereço armazenado em **p** será alterado.
- Veja o código: `remove_final.c`

Como liberar uma lista



- `free(p)` libera apenas o primeiro nó;
- é necessário percorrer a lista liberando todos os nós;
- Veja o código: `libera.c`