ESTRUTURAS DE DADOS

Prof. H. Senger

1 Listas

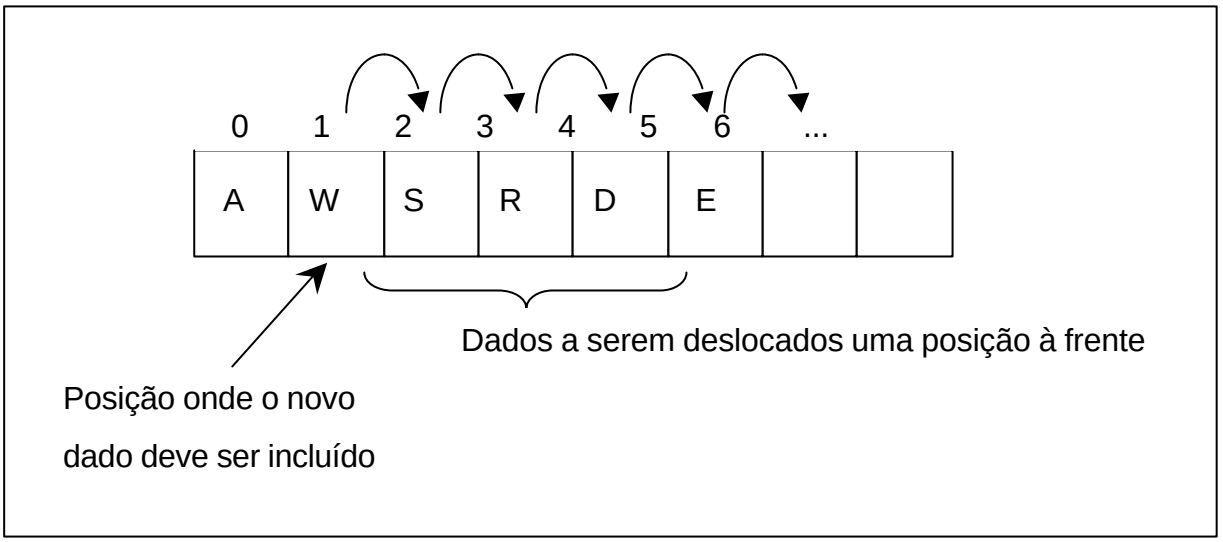
Existem duas maneiras bastante utilizadas na construção de **listas**.

- Utilizando vetores
- Utilizando alocação dinâmica de memória

2 Implementação de LISTAS usando vetores

A implementação de **listas** baseada em vetores traz alguns problemas:

⇒ Necessidade de **deslocar** os dados existentes, cada vez que um dado tiver de ser removido ou incluído na lista. No exemplo abaixo, se quisermos inserir um novo dado na posição 1 do vetor, é preciso deslocar para frente todos os dados que estão a partir dessa posição, até o fim.



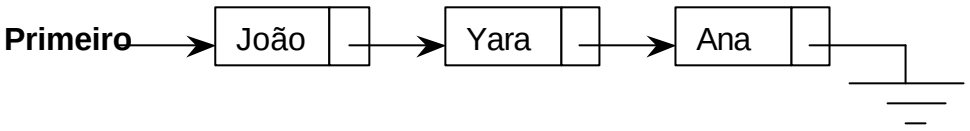
⇒ **Tamanho fixo**: Depois que o programa foi compilado e iniciou a execução, não é mais possível mudar o tamanho do vetor.

Solução : Implementar de outra forma. Se possível, de um modo que

- Permita acrescentar e remover dados, consumindo a quantidade de memória exata (só o necessário, nem mais, nem menos)
- Não exija o deslocamento dos dados a cada inserção/remoção

3 Implementação de LISTAS usando alocação dinâmica

De agora em diante, os dados da lista não mais serão guardados em um vetor, mas em **células** com o seguinte aspecto :



Note que cada célula contém um dado, e aponta (isto é, indica) qual é a próxima célula.

3.1 Criação de uma lista

O algoritmo abaixo fica constantemente incluindo novos dados na lista :

```
Loop                                    /* loop infinito */
    Ler_teclado (novo_dado);
    Criar (nova_célula);
    Nova_célula ← novo_dado;  /* guarda novo_dado em nova_celula*/
    Atualiza( Primeiro);
Fim-loop
```

3.2 Destruição da lista

À medida que o programa não precise mais de um dado da lista, ele pode removê-lo. O algoritmo abaixo fica continuamente retirando dados da lista, destruindo a célula de onde o dado foi retirando e liberando a memória:

```
Loop                    /* loop infinito */
    Dado ← conteúdo da célula;  /* guarda o conteúdo da primeira célula*/
    Utiliza(dado);              /* faz alguma coisa com o dado ... */
    Segundo ← Próximo_de (Primeiro); /* guarda quem é o segundo */
```

Destrói_celula_indicada_por (Primeiro);

Primeiro ← Segundo; */*Atualiza o indicador de início da lista*/*

Fim-loop

PROBLEMA:

Imagine como construir um programa com as seguintes características :

- tem de armazenar (na memória) uma grande quantidade de dados
- esse programa será executado juntamente com outros programas no mesmo computador, e portanto, ele só deve "Alocar" (reservar, usar) a quantidade de memória realmente necessária
- em algum momento, o usuário pode pedir para
  - ✓ digitar mais dados
  - ✓ apagar alguns dados existentes

SOLUÇÃO:

Esse programa deveria possuir o seguinte algoritmo :

Loop */* loop infinito */*

Lê_teclado_ou_mouse (comando);

Se (comando = "Incluir") então

Início

Ler_teclado (novo_dado);

Criar (nova_célula);

Nova_célula ← novo_dado; */* guarda novo_dado */*

Atualiza(Primeiro);

Fim-se

Senão Se (comando = "Apagar") então

Início

Segundo ← Próximo_de (Primeiro); */* guarda end. do segundo */*

Destrói_celula_indicada_por (Primeiro);

Primeiro ← Segundo; */*Atualiza o indicador de início da lista*/*

Fim-se;

Fim-loop