Aula 3

Estrutura de Dados

Prof. Vinicius Pozzobon Borin

1

Conversa Inicial

2

- **Está na hora de conhecer e aprender uma nova estrutura de dados**

3

- **Listas encadeadas**
- **Pilhas (*stacks*)**
- **Filas (*queues*)**

4

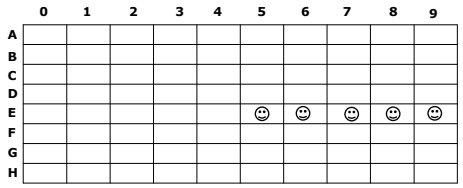
***Arrays* e**
listas encadeadas

5

- ***Arrays* ou vetores**
  - **São estruturas de dados que trabalham com alocação de dados sequenciais na memória**

6

O problema dos assentos reservados no cinema

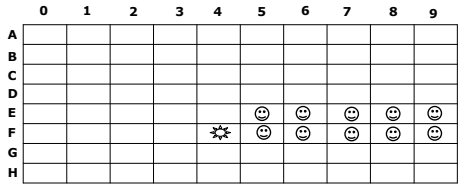


7

- ***Array* estático**
  - **É um conjunto de espaços sequencial na memória que é previamente reservado, mas que em hipótese alguma pode ter o seu tamanho alterado**

8

O problema dos assentos sem reserva no cinema

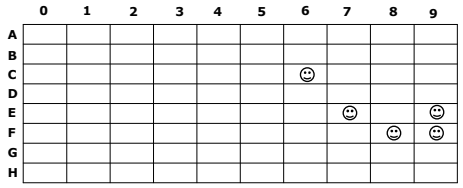


9

- ***Array* dinâmico**
  - **Armazena dados sequencialmente, porém agora, à medida que surgem novos dados, podemos realocar o conjunto em outro espaço de memória em que caibam todos os dados**

10

O problema dos amigos distantes no cinema



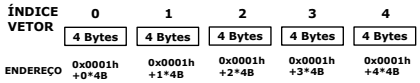
11

- **Listas encadeadas**
  - **Arranjo de dados sem necessidade de estarem sequencialmente alocados**

12

Arrays

- **Conjuntos de dados alocados sequencialmente na memória do programa**



13

Arrays

- **O tempo de acesso aos dados é constante e independe do tamanho do conjunto de dados**

14

Arrays

- **Estáticos**
  - **Alocam um bloco de memória que fica alocado sempre, independentemente do seu uso**
- **Dinâmicos**
  - **Alocam somente o espaço sendo usado**

15

Atenção!

16

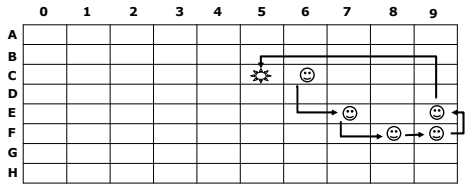
Listas encadeadas

- **Alocação não sequencial**
- **Dados esparsos na memória do programa**

17

- **Se os dados estão esparsos na memória, como a estrutura de lista encadeada localiza seus elementos?**
  - **Cada elemento da lista encadeada armazena na memória não só seus dados, como também o endereço onde está localizado o próximo elemento na memória**

18



19

- Se a lista encadeada é tão boa assim para alocação de memória, por que as linguagens ainda insistem em trabalhar com *arrays*?
  - Em uma lista encadeada, cada elemento conhece somente o próximo elemento, pois armazena somente o endereço deste

20

O problema do livro sem paginação

- Livro paginado - *array*
- Livro não paginado - lista encadeada

21

Complexidades

Função	*Array*	Lista encadeada
Leitura	O(1)	O(n)
Inserção no início	O(n)	O(1)
Inserção no fim	O(n)	O(n)
Inserção no meio	O(n)	O(n)

22

Construindo listas encadeadas

23

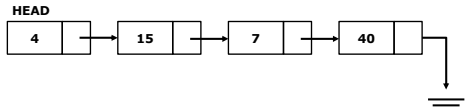
- Também conhecida como
  - Lista ligada
  - *Linked list*
- Cada elemento da lista é um
  - Nó ou nodo

24

- **Cada elemento da lista é composto, portanto, minimamente de duas informações: o(s) dado(s) a ser(em) armazenado(s) e um endereço (ponteiro) na memória do próximo elemento da lista encadeada**

25

Representação da lista encadeada



26

- **Em suma as características das listas encadeadas são**
  - **Sucessivos elementos conectados por ponteiros**
  - **O último elemento aponta para um endereço nulo, o que caracteriza o final da lista**
  - **Funciona dinamicamente, aumentando e diminuindo de tamanho conforme necessita**

27

- **Pode utilizar toda a memória destinada ao programa, uma vez que cada dado pode ficar isolado na memória**
- **Não desperdiça memória, alocando somente o que precisa**
- **Apresenta tempos de leitura Big-O de dados inferior aos *arrays***

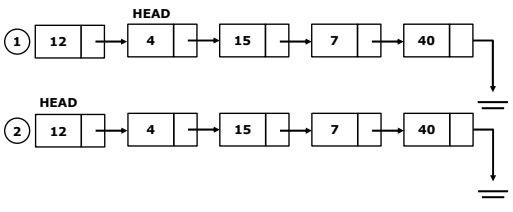
28

Listas encadeadas simples

- **Vejamos a implementação em Python**

29

Inserção no início da lista encadeada



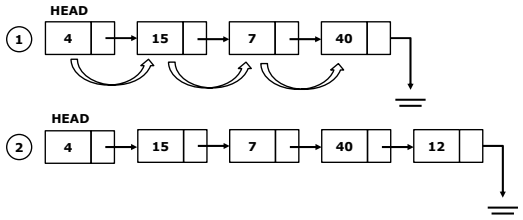
30

Inserção no início da lista encadeada

- Vejamos a implementação em código

31

Inserção no final da lista encadeada



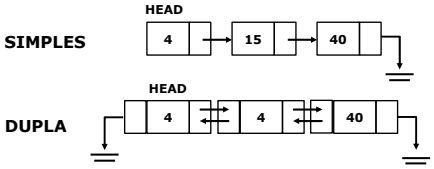
32

Inserção no final da lista encadeada

- Vejamos a implementação em código

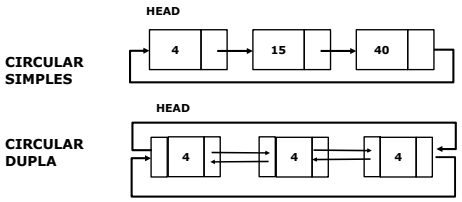
33

Classificação de listas encadeadas



34

Classificação de listas encadeadas



35

Pilhas (*stacks*)

36

A pilha de anilhas de academia

Ilja Generalov/shutterstock

37

- Uma estrutura de dados é uma pilha quando só conseguimos manipular o que está no seu topo
- O primeiro que entra é o último que sai
- *First in last out* (FILO)

38

Exemplos de aplicações

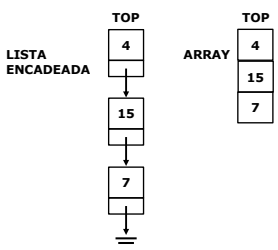
- Recursividade
- Cálculo de expressões matemáticas
- Jogo FreeCell

39

Construindo e manipulando uma pilha

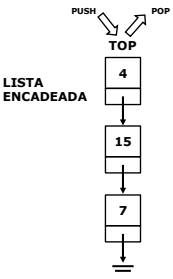
- O top é a única posição que conseguimos manipular
- Podemos implementar uma pilha com listas encadeadas ou com *arrays*

40



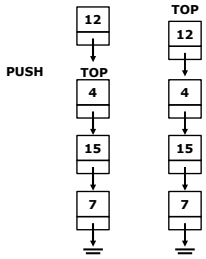
41

- Empilhar - *push*
- Desempilhar - *pop*



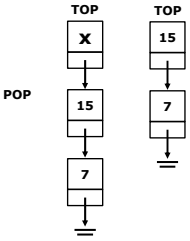
42

Empilhando (*push*)



43

Desempilhando (pop)



44

Filas (*queues*)

45

A fila do supermercado

46

- **Uma estrutura de dados é uma fila quando inserimos somente no final dela e removemos somente do seu início**
- **O primeiro que entra é o primeiro que sai**
- ***First in first out* (FIFO)**

47

Exemplos de aplicações

- **Fila de impressão**
- **Lista de reprodução de música**
- **Jogo Snake**

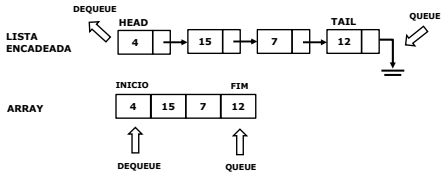
48

Construindo e manipulando uma fila

- **Inserimos no final (*tail*) e removemos do início (*head*)**
- **Podemos implementar uma pilha com listas encadeadas ou com *arrays***

49

- **Enfileirar - *queue***
- **Desenfileirar - *dequeue***



50

Implementando pilhas e filas

51

- **Implementação com *arrays* dinâmicos (listas em Python)**

52

Código da pilha

- **Vamos diretamente no código?**

53

Código da fila

- **Vamos diretamente no código?**

54

Referências

55

- **ASCENCIO, A.F.G; ARAÚJO, G.S. Estruturas de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010.**
- **BHARGAVA, A. Y. Entendendo algoritmos. São Paulo: Novatec, 2017.**
- **DROZDEK, A. Estrutura de dados e algoritmos em C++. São Paulo: Cengage Learning Brasil, 2018.**

56

- **FERRARI, R. et al. Estruturas de dados com jogos. [s.l]: Elsevier, 2014.**
- **KOFFMAN, E.B.; WOLFGANG, P.A.T. Objetos, abstração, estrutura de dados e projeto usando C++. São Paulo: Grupo GEN, 2008.**

57