ESTRUTURA DE DADOS

Lucas Plautz Prestes

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Listas sequenciais: estáticas e dinâmicas

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Reconhecer as características de implementação de listas através de estruturas estáticas (vetores).
- Definir as características de implementação de listas através de estruturas dinâmicas (listas encadeadas).
- Apresentar exemplos de listas estáticas e dinâmicas.

Introdução

A compreensão de listas em programação é de fundamental importância para a correta manipulação de dados em memória. A utilização de listas, sejam elas estáticas ou dinâmicas, facilita a manipulação de dados, pois a partir de um único identificador será possível armazenar uma infinidade de informações do mesmo tipo, evitando a necessidade de criar uma variável para cada informação.

Atualmente, realizamos listas para diversas ações em nosso dia a dia que abrangem desde uma simples lista de compras — chamada de classe — a tarefas a serem cumpridas em nossas profissões. Logo, se as listas fazem parte do nosso cotidiano, no qual criamos *software* para facilitar as ações diárias, nada mais natural do que haver um modo de representá-las em processos de automatização e manipulação de dados em programação.

Neste capítulo, você vai estudar sobre o processo de manipulação de listas estáticas e dinâmicas desde os seus conceitos básicos até a sua implementação em linguagem Python.

1 O que são listas?

Na computação, uma lista pode ser definida como um conjunto de elementos do mesmo tipo, agrupados e identificados por um identificador único e separados entre si em "caixas", chamados de nodos, que ocupam um endereço específico na memória. O relacionamento entre os nodos é definido por sua posição em relação aos demais nodos, assim como pessoas em uma fila, porém indexados na memória do computador.

Toda lista apresenta um nodo inicial, o primeiro elemento da lista. A partir deste seguirá uma sequência de nodos conforme uma ordem predefinida pelo programador, assim como uma fila de banco. Todos os nodos de uma lista têm, geralmente, o mesmo tipo de dado, podendo ser do tipo primitivo, como um inteiro ou abstrato, criado pelo programador. Na Figura 1 é possível verificar de forma ilustrada um exemplo de nodo com a sua respectiva posição.

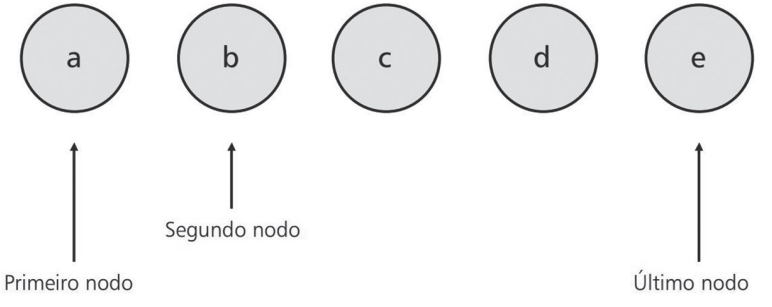


Figura 1. Lista composta por cinco elementos, nodos, sendo cada um identificado por uma letra do alfabeto.

Fonte: Edelweiss e Galante (2009, p. 50).

Exemplo

Hoje é o quinto dia útil do mês e João foi ao banco realizar o pagamento de seu aluguel. Ao chegar, João verificou que havia 30 cadeiras numeradas de 1 a 30, nas quais as pessoas se sentavam na ordem de chegada. Como o objetivo é relacionarmos situações reais com o desenvolvimento de listas, devemos renumerá-las com as posições de

um vetor, no qual a primeira posição deverá assumir o número 0, e a última posição, o número 29, totalizando as 30 posições.

A partir dessa análise, João logo pensou em qual era a relação da fila do banco com as listas apreendidas em sala de aula. As cadeiras são os nodos, e os números, a sua respectiva localização no banco, em um total de 30 lugares numerados de 0 a 29. As pessoas sentadas são o conteúdo de cada nodo, sendo do tipo abstrato, criado pelo programador.

Podemos definir alguns exemplos no uso de listas no cotidiano adaptados de Edelweiss e Galante (2009, p. 50), como veremos a seguir.

- As notas individuais dos alunos em uma turma, ordenadas de forma alfabética de acordo com o seu respectivo nome.
- Cadastros de funcionários em uma determinada empresa, organizados, primeiramente, de acordo com o seu respectivo setor e, posteriormente, de acordo com o seu respectivo número de inscrição no momento em que foi admitido na empresa.
- Os respectivos dias de uma semana, nos quais domingo se refere ao primeiro dia, e os demais seguem na respectiva ordem, em um total de sete elementos.
- Valores obtidos por meio de medições em uma indústria, tais como temperatura e pressão de motores industriais ordenados de acordo com a data e a hora que foram coletados.

2 Estruturas estáticas — vetores e matrizes

Uma estrutura estática pode ser definida como sendo uma estrutura de dados primariamente unidimensional que armazena uma quantidade de dados predeterminados em nodos do mesmo tipo, em uma mesma variável, de forma sequencial em memória. O vetor é um exemplo de estrutura estática, em que cada posição tem um índice referente à sua localização, localizador do nodo, no qual cada posição é capaz de armazenar um determinado dado (Figura 2).

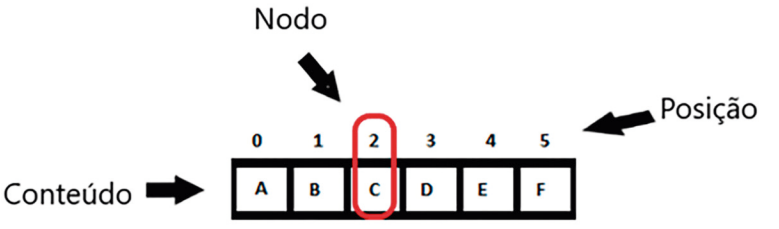


Figura 2. Vetor composto por seis posições, nodos, localizados nos índices numerados de 0 a 5. O conteúdo do vetor é do tipo char, composto pelas letras A, B, C, D, E, F.

A seguir, vemos as vantagens de se utilizar vetores.

- Fácil implementação comparada a listas encadeadas.
- Redução da quantidade de memória utilizada, pois não requer armazenar a localização do próximo nodo.
- A sua dimensão está disponível sem a necessidade de percorrer todos os nodos.

Entretanto, há desvantagens na utilização de vetores, como mostrado a seguir.

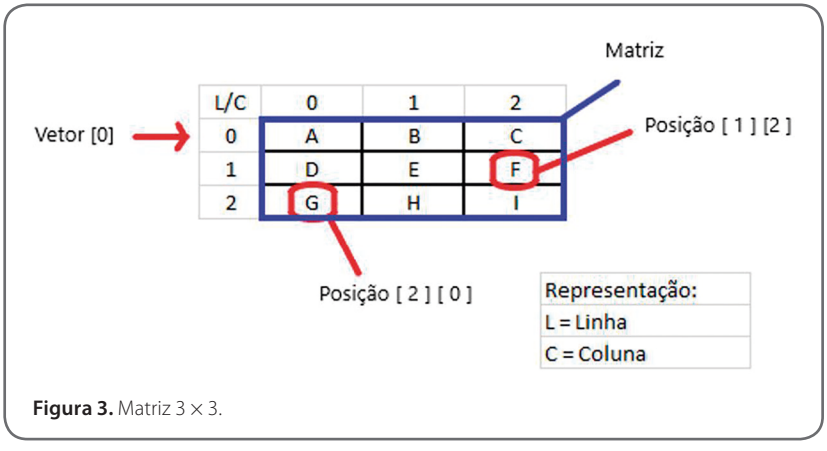
- Dificuldade na ordenação dos seus elementos.
- O aumento do seu tamanho poderá requerer o deslocamento do vetor em memória, caso não haja espaço de forma linear.
- A impossibilidade de adicionar um elemento no meio do vetor sem realizar o deslocamento de todos os demais nodos, logo, consumindo alto processamento.

Fique atento

Geralmente, ao utilizarmos vetores, devemos declarar o tamanho e o tipo de dado que serão armazenados em cada nodo, porém, dependendo da linguagem de programação escolhida, essa premissa poderá ser falsa, como na linguagem Python.

Ao aprendermos a utilizar vetores, indiretamente estamos compreendendo a estrutura de uma matriz, que é muito utilizada no ensino de matemática e aplicada de forma extensiva em análise e controle de equipamentos nos cursos de engenharia, bem como na área de inteligência artificial no processamento de imagens em algoritmos de identificação de objetos.

Não muito diferente do vetor, a matriz nada mais é que uma lista de vetores, na qual cada linha representa um respectivo vetor e cada posição da matriz tem uma identificação única. Na Figura 3 e no quadro a seguir, podemos identificar as posições de uma matriz.



L/C	0	1	2
0	A	B	C
1	D	E	F
2	G	H	I

Figura 3. Matriz 3 × 3.

Conteúdo	Posição na matriz [linha][coluna]
A	[0][0]
B	[0][1]
C	[0][2]
D	[1][0]
E	[1][1]
F	[1][2]
G	[2][0]
H	[2][1]
I	[2][2]

3 Estruturas dinâmicas — listas encadeadas

As estruturas dinâmicas são utilizadas para relacionar itens e nodos que precisam ser manipulados em tempo de execução com dimensão indefinida. A partir dessa premissa, podemos dizer que durante o tempo de execução será possível adicionar ou remover itens de uma estrutura dinâmica de dados de acordo com a necessidade do usuário. Um exemplo de utilização de lista dinâmica de dados poderá ser representado por um cadastro de usuários em um sistema acadêmico, que frequentemente deverá ser modificado com a adição e a remoção de alunos (SCHILDT, 1996).

As estruturas dinâmicas podem ser representadas conceitualmente a partir de listas encadeadas, nas quais têm alguns conceitos necessários para a sua compreensão. Estes elementos são:

- conteúdo — representa o dado armazenado em uma posição de memória, podendo ser do tipo primitivo ou criado pelo usuário;
- ponteiro — representa uma posição, apontamento, específica da memória e, neste caso, representa a posição do próximo elemento de uma lista;
- nodo — é representado por um espaço de memória que contém um conteúdo e um ponteiro. O conteúdo representa o dado do respectivo nodo, e o ponteiro, a posição de memória na qual se encontra o próximo nodo. A Figura 4 ilustra a representação de um nodo.

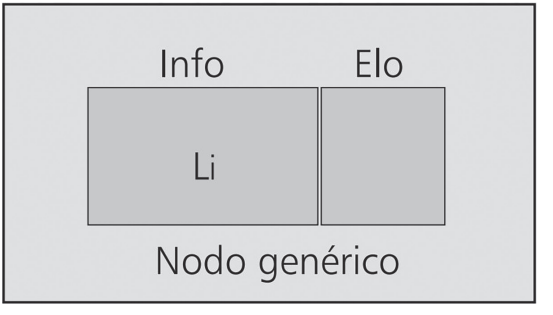


Figura 4. Nodo de uma lista encadeada simples.
Fonte: Edelweiss e Galante (2009, p. 89).

As listas dinâmicas, por terem nodos compostos de dados e ponteiros, poderão ser a opção mais adequada para a manipulação de uma grande quantidade de dados, pois, a partir da possibilidade de manipulação dos ponteiros, serão realizadas ações de inserção e de remoção de elementos e nodos sem a necessidade de reordenação ou manipulação de uma grande massa de dados, como ocorre nos vetores.

A seguir, encontram-se algumas ações realizadas em listas dinâmicas de dados.

- Inserção de um novo nodo: a inserção poderá ser realizada em qualquer posição da lista sem a necessidade de reordenação de todos os elementos. Durante a inserção, somente é necessário readequar os respectivos ponteiros.
- Remoção de um nodo: a remoção de um novo nodo poderá ser realizada em qualquer local da lista dinâmica sem a necessidade de reestruturação de todos os elementos dela. Ao remover um nodo, será necessário realizar a readequação dos ponteiros impactados na sua remoção.
- Ordenação de elementos: a lista dinâmica possibilita a realização da ordenação dos seus elementos durante a inserção dos nodos, não havendo a necessidade de realizar a ordenação posteriormente à sua criação. Caso haja a necessidade de reordenação posterior à sua criação, poderão ser aplicadas diferentes técnicas para a sua realização, com menor consumo de processamento se comparado a vetores.

As listas dinâmicas são essencialmente representadas por listas encadeadas, sejam elas simples ou duplas.

- Lista encadeada simples: a lista encadeada simples se refere a um conjunto de nodos interligados a partir de seus ponteiros. Cada nodo tem o seu respectivo conteúdo, dado, bem como o endereço de memória do próximo nodo. Caso a lista encadeada simples seja linear, o último nodo terá apontamento nulo, pois não haverá nodo a ser referenciado. A lista encadeada simples poderá ser de caráter circular, na qual o último nodo irá apontar para o primeiro nodo da lista. A Figura 5 ilustra uma lista encadeada linear.

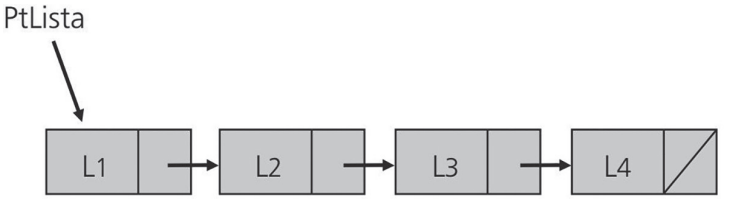


Figura 5. Lista encadeada simples de caráter linear.
Fonte: Edelweiss e Galante (2009, p. 89).

- Lista duplamente encadeada: a lista duplamente encadeada contém um conjunto de nodos interligados entre si a partir de seus ponteiros. Diferentemente da lista encadeada simples, a lista dupla contém dois ponteiros para cada nodo. Estes ponteiros referenciam os nodos posteriores e inferiores ao nodo de referência. O primeiro nodo tem o seu ponteiro inferior nulo, assim como o ponteiro posterior do último nodo da respectiva lista. Caso a lista duplamente encadeada tenha característica circular, o primeiro e o último nodo irão referenciar-se em seus respectivos ponteiros, como ilustrado na Figura 6.

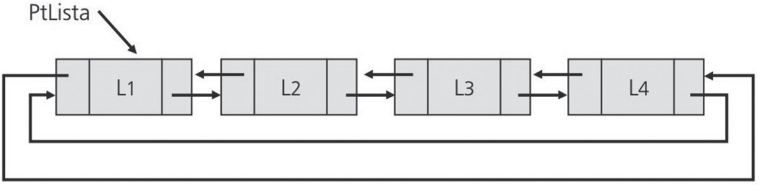


Figura 6. Lista duplamente encadeada circular.
Fonte: Edelweiss e Galante (2009, p. 110).

Fique atento

Ao trabalharmos com listas circulares, será necessário um indicador que direcione para o início da lista. Na Figura 6, há um indicador referenciando o primeiro elemento da lista, o nodo L1.

Vantagens da utilização de listas dinâmicas:

- tamanho indefinido de uma lista;
- possibilidade de adição e remoção de elementos em qualquer posição da lista;
- melhor utilização da memória do computador, em que a posição alocada em memória não precisa ser obrigatoriamente linear;
- a ordenação dos seus elementos poderá ser realizada de forma mais rápida se comparada com uma lista estática.

Desvantagens da utilização de listas dinâmicas:

- maior processamento na adição de novo nodo;
- maior consumo de memória em cada nodo por haver a necessidade de armazenar a posição do próximo nodo;
- há a necessidade de percorrer toda a lista para saber o seu tamanho.

4 Exemplos de listas — estática e dinâmica

Exemplos de declaração de um vetor, atribuição de conteúdo e impressão em Python adaptados de Lutz (2011):

Código em Python:

```
#Inicio
#Declaração de variável - Vetor
funcionario = ['nome','idade','salario']
joao = ['joao', 42, 3000]
maria = ['Maria', 45, 4000]
media = 0
i = 0

# impressão de conteúdo em posição específica
print('alunos',joao[0], 'e', maria[0])
print('Idade', joao[1], 'e', maria[1])
print('Renda', joao[2], 'e', maria[2])
```

```
# Impressão do vetor completo

print(funcionario)


# Atribuição de conteúdo em posição específica do vetor

funcionario[0] = 'Lucas'
funcionario[1] = 18
funcionario[2] = 5000



# Cálculo da média de salários
media = (funcionario [2] + joao [2] + maria [2] )/ 3


# Impressão da média de salários

print('A media dos salarios e de :', media)

#Fim
```

Saída do programa:

```
alunos joao e Maria
Idade 42 e 45
Renda 3000 e 4000
['nome', 'idade', 'salario']
A media dos salarios e de : 4000.0
```

Fique atento

Durante o desenvolvimento de vetores, devemos tomar cuidado para não acessar uma posição do vetor inexistente. Caso isso ocorra, o erro a seguir deverá aparecer na sua tela:

```
IndexError: list index out of range
```

Exemplos de criação e manipulação de uma matriz em Python realizada pelo autor:

Código em Python:

```
# Inicio
#Declaração de uma Matriz
matriz = [['A', 'B', 'C', 'D'],
  ['E', 'F', 'G', 'H'],
  ['i', 'j', 'k', 'l']]

# Impressão de todo o vetor
print("Matriz =", matriz)

#Impressão de cada vetor pertencente a matriz
print("vetor[0] =", matriz[0])
print("vetor[1] =", matriz[1])
print("vetor[2] =", matriz[2])

#Impressão de uma posições específicas de uma matriz

print("posicao[1][2] =", matriz[1][2])
print("posicao[0][-1] =", matriz[0][-1])

#Fim
```

Saída do programa:

```
Matriz = [['A', 'B', 'C', 'D'], ['E', 'F', 'G', 'H'], ['i', 'j', 'k',
'l']]
vetor[0] = ['A', 'B', 'C', 'D']
vetor[1] = ['E', 'F', 'G', 'H']
vetor[2] = ['i', 'j', 'k', 'l']
posicao[1][2] = G
posicao[0][-1] = D
```

Exemplos de utilização da biblioteca de lista em Python adaptados de Borges (2010):

Código em Python:

```
#inicio

# Uma nova lista
generos = ['policial', 'indefinido', 'terror', 'suspense', 'amor']

# Varrendo a lista
for prog in generos:
 print (prog)

# Trocando o último elemento
generos[-1] = 'romantico'
# Incluindo
generos.append('drama')

# Removendo
generos.remove('indefinido')

# Imprime numerado
for i, prog in enumerate(generos):
 print (i + 1, '=>', prog)


print (generos)

#Fim
```

Saída do programa:

```
policial
indefinido
terror
suspense
amor
1 => policial
2 => terror
3 => suspense
4 => romantico
5 => drama
['policial', 'terror', 'suspense', 'romantico', 'drama']
```

O estudo inicial relacionado às listas, sejam elas estáticas ou dinâmicas, com as suas respectivas vantagens e desvantagens, nos traz questionamentos:

Qual lista devo utilizar durante o desenvolvimento de uma aplicação?

A resposta é simples, depende do que você está desenvolvendo e qual é o seu objetivo na manipulação dos dados armazenados. Se você tem um número definido e imutável de elementos armazenado e não tem a necessidade de ordenação, o vetor poderá ser a melhor opção. Caso você não saiba a quantidade de dados que serão armazenados e, durante a sua manipulação, seja necessário algum tipo de ordenação dos elementos, a lista dinâmica será a melhor opção.

A partir dos estudos realizados neste capítulo, foi possível compreender as diferenças entre os tipos de listas, sejam elas estáticas ou dinâmicas, com as suas respectivas particularidades. Logo, esse é o primeiro passo para a compreensão e a manipulação de estruturas de dados de pequeno e grande portes a partir da disponibilidade massiva de dados na era digital.

Referências

BORGES, L. E. *Python para desenvolvedores*. 2. ed. Rio de Janeiro, Edição do Autor, 2010.

EDELWEISS, N.; GALANTE, R. *Estruturas de dados*. Porto Alegre: Bookmann, 2009.

LUTZ, M. *Programming Python*. 4th ed. Beijing: O'Reilly, 2011.

SCHILDT, H. *C*: completo e total. 3. ed. São Paulo: Makron Books, 1996.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS