CENTRO UNIVERSITÁRIO FEI
SILVIO ROMERO DE ARAÚJO JÚNIOR

Listas Ligadas

São Bernardo do Campo
2019

SILVIO ROMERO DE ARAÚJO JÚNIOR

Listas Ligadas

Relatório de atividade para a disciplina Programação Científica - PEL216.

São Bernardo do Campo

2019

RESUMO

O estudo de estruturas de dados serve como base para vários campos da ciência da computação, como programação, projeto de compiladores e gerenciamento de banco de dados. Quase todos os programas usam estruturas de dados como um meio eficaz de armazenamento e organização de dados. Muitas vezes, o sucesso de um programa depende da maneira como os dados são representados e do algoritmo usado para processar os dados. Uma estrutura de dados usa uma coleção de variáveis relacionadas que podem ser acessadas individualmente ou como um todo e representa um conjunto de itens de dados com um relacionamento específico entre eles. Assim, a escolha de uma estrutura de dados efetiva é a chave para o sucesso no projeto de algoritmos. E as estruturas conhecidas como listas ligadas (ou encadeadas), figuram entre as mais importantes neste campo de estudo.

Palavras-chave: algoritmos, estrutura de dados, listas ligadas, listas encadeadas.

ABSTRACT

The study of data structures serves as the basis for various fields of computer science, such as programming, compiler design, and database management. Almost all programs use data structures as an effective and efficient means of storing and organizing data. Often, the success of a program depends on how the data is represented and the algorithm used to process the data. A data structure uses a collection of related variables that can be accessed individually or as a whole and represents a set of data items with a specific relationship between them. Thus, choosing an effective data structure is the key to success in algorithm design. And the structures known as linked lists, are among the most important in this field of study.

Keywords: algorithms, data structures, linked lists.

SUMÁRIO

1 INTRODUÇÃO

No contexto de sistemas computacionais, estruturas de dados possuem aplicações nas mais diversas áreas, como por exemplo programação, sistemas de banco de dados, redes de computadores e inteligência artificial. Uma estrutura de dados é basicamente um grupo de elementos de dados que são reunidos sob um nome, definindo um modo particular de armazenamento e organização, levando em consideração o arranjo lógico e matemático na memória de um computador para que possam ser usados com eficiência.

A escolha das estruturas de dados e algoritmos apropriados forma a etapa fundamental no projeto de um programa eficiente. Assim, sua compreensão é essencial para os pesquisadores e estudantes que desejam trabalhar na concepção e implementação de sistemas de Software. Este relatório enfoca os tipos de estrutura de dados conhecidos listas ligadas.

1.1 OBJETIVO

Este trabalho tem como objetivo apresentar uma proposta de implementação da estrutura de dados conhecida como listas ligadas, utilizando a linguagem de programação C++. Com o intuito de avaliar as limitações das estruturas de dados estáticas e compreender como criar um programa onde a memória pode diminuir e crescer dinamicamente.

1.2 MOTIVAÇÃO

O estudo de estrutura de dados é parte essencial dos problemas computacionais, as listas ligadas fazem parte das estruturas compostas e lineares de amplo uso em tais problemas e com aplicações cruciais no desenvolvimento de aplicações. Além disso, as listas ligadas tentam contornar os problemas comuns existentes nos vetores, como por exemplo a reserva de memórias em espaços adjacentes. Sendo assim, um estudo pormenorizado de seus conceitos e como implementá-las se faz necessário. Um programa de computador deve produzir resultados corretos obviamente, porém deve fazê-lo de forma eficiente em termos de espaço de memória executado e tempo de execução.

1.3 METODOLOGIA

Primeiramente foi efetuada uma pesquisa bibliográfica com base em livros abordando os conceitos e terminologia aplicada ao presente trabalho. Após esta primeira etapa, avaliou-se um algoritmo genérico para listas ligadas e suas operações, visando a compreensão necessária para codificação de um programa em linguagem C++, com base no paradigma de programação orientada a objetos.

2 REVISÃO DA LITERATURA

Uma lista ligada é uma estrutura de dados que pode armazenar uma quantidade indefinida de itens. Esses itens são conectados usando ponteiros de maneira sequencial. Existem dois tipos de lista ligada; lista com link único e lista duplamente ligada. Em uma lista única, cada elemento contém alguns dados e um link para o próximo elemento. Por outro lado, cada nó em uma lista duplamente ligada contém alguns dados, um link para o próximo nó e um link para o nó anterior.

Os elementos de uma lista ligada são chamados de nós. Um nó tem dois campos, como por exemplo, “dados” e “próximo elemento”. O campo de “dados” contém os dados que estão sendo armazenados nesse nó específico. Não pode ser apenas uma única variável. Pode haver muitas variáveis apresentando a seção de dados de um nó. O campo “próximo elemento” contém o endereço do próximo nó. Então este é o lugar onde o link entre nós é estabelecido.

2.1.1 Operações em Listas Ligadas

Segundo [THAREJA, 2014], as operações primitivas executadas na lista ligada são as seguintes, os respectivos algoritmos para elas também são apresentados (quando aplicável):

2.1.1.1 Criação

A operação de criação é usada para criar uma lista ligada. Depois que uma lista ligada é criada com um nó, a operação de inserção é usada para adicionar mais elementos neste nó.

2.1.1.2 Inserção

A operação de inserção é usada para inserir um novo nó em qualquer local especificado na lista ligada. Um novo nó pode ser inserido:

(a) No início da lista ligada

```
Step 1: IF AVAIL = NULL
                Write OVERFLOW
                Go to Step 7
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE ->  DATA = VAL
Step 5: SET NEW_NODE ->  NEXT = START
Step 6: SET START = NEW_NODE
Step 7: EXIT
```

(b) No final da lista ligada

```
Step 1: IF AVAIL = NULL
                Write OVERFLOW
                Go to Step 1
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE ->  DATA = VAL
Step 5: SET NEW_NODE ->  NEXT = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR -> NEXT != NULL
Step 8: SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 9: SET -> PTR NEXT = NEW_NODE
Step 10: EXIT
```

(c) Em qualquer posição especificada entre uma lista ligada

```
Step 1: IF AVAIL = NULL
                Write OVERFLOW
                Go to Step 12
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE -> DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PREPTR -> DATA != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 10: PREPTR -> NEXT = NEW_NODE
Step 11: SET NEW_NODE -> NEXT = PTR
Step 12: EXIT
```

2.1.1.3 Remoção

A operação de remoção é usada para excluir um item (ou nó) da lista ligada. Um nó pode ser excluído do:

(a) Início de uma lista ligada

```
Step 1: IF START = NULL
                Write UNDERFLOW
                Go to Step 5
            [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START -> NEXT
Step 4: FREE PTR
Step 5: EXIT
```

(b) Fim de uma lista ligada

```
Step 1: IF START = NULL
                Write UNDERFLOW
                Go to Step 8
        [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != NULL
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 6: SET PREPTR -> NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

c) Localização especificada da lista ligada

```
Step 1: IF START = NULL
                Write UNDERFLOW
                Go to Step 10
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Steps 5 and 6 while PREPTR -> DATA != NUM
Step 5: SET PREPTR = PTR
Step 6: SET PTR = PTR NEXT
[END OF LOOP]
Step 7: SET TEMP = PTR
Step 8: SET PREPTR -> NEXT = PTR -> NEXT
Step 9: FREE TEMP
Step 10 : EXIT
```

2.1.1.4 *Examinar*

Examinar é o processo de passar por todos os nós de um extremo a outro de uma lista ligada. Em uma lista unicamente ligada, podemos visitar da esquerda para a direita, e para frente.

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3:         Apply Process to PTR -> DATA
Step 4:         SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 5: Write COUNT
Step 6: EXIT
```

2.1.1.5 *Pesquisar*

É o processo de localizar um determinado nó dentro de uma lista ligada.

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Step 3 while PTR != NULL
Step 3:         IF VAL = PTR -> DATA
                        SET POS = PTR
                        Go To Step 5
                ELSE
                        SET PTR = PTR -> NEXT
                [END OF IF]
        [END OF LOOP]
Step 4: SET POS = NULL
Step 5: EXIT
```

2.1.1.6 *Concatenação*

Concatenação é o processo de anexar a segunda lista ao final da primeira lista. Considere uma lista A com n nós e B com m nós. Em seguida, a concatenação da operação colocará o primeiro nó de B no nó (n + 1) th em A. Após a concatenação, A conterá nós (n + m).

3 IMPLEMENTAÇÃO PROPOSTA

Na implementação do código proposto no presente relatório utilizou-se os seguintes recursos de Hardware:

- MacBook Pro (Retina, 15-inch, Early 2013);
- Processador: 2,4 GHz Intel Core i7;
- Memória: 8 GB 1600 MHz DDR3.

Com relação aos softwares utilizados, citam-se:

- macOS Mojave Versão 10.14.5
- Xcode 10.2.1

Conforme requisitos da atividade as estruturas de dados foram criadas como classes (conforme códigos em C++ anexados).

Os algoritmos para as operações mais comuns em listas ligadas foram aprendizados no item **2.1.1 (Operações em Listas Ligadas).**

4 EXPERIMENTOS E RESULTADOS

Depois de consultados os algoritmos por meio das referências bibliográficas, foram desenvolvidos dois códigos em linguagem C++ para execução dos experimentos, um para a pilha e outro para fila. Utilizou-se primeiramente, uma capacidade pequena para execução dos programas (um tamanho máximo de capacidade de três elementos para as duas estruturas de dados, definidos pela constante TAM no início do programa). Depois, incrementou-se o tamanho da capacidade de armazenamento de elementos para verificar se o programa funciona corretamente para números distintos de elementos. Tal adaptação é muito simples, necessitando alterar apenas o valor da constante TAM no início do programa. Atingindo-se assim os resultados esperados, que consistiu em visualizar o comportamento dos algoritmos e seu correto funcionamento.

5 TRABALHOS CORRELATOS

Muitos autores têm abordado o tema de listas ligadas (DAS, 2006, SAHNI, 2005, THAREJA, 2005). Porém, não foram encontradas diferenças significativas do ponto de vista de implementação das estruturas de dados abordadas, uma razão possível é o fato do tema já estar bem consolidado no meio acadêmico. Sendo assim a pesquisa com intuito de comparação resultou em uma confirmação das ideias gerais de elaboração dos algoritmos.

6 CONSIDERAÇÕES FINAIS

Este relatório apresentou os conceitos teóricos sobre listas ligadas. Viu-se também as operações envolvidas para manipulação de tais estruturas. Uma implementação que usa ponteiros e alocação de memória dinâmica é chamada de lista ligadas. Estas estruturas de dados dinâmicas são muito eficientes, pois os itens podem ser adicionados ou excluídos em qualquer posição com muita facilidade em comparação com vetores. Possuindo diversas aplicações em programação científica e sendo preferidas quando necessita-se de alocação dinâmica de memória.

REFERÊNCIAS

DAS, V. V. **Principles of Data Structures Using C and C++**. New Delhi, India. New Age International (P) Ltd., Publishers, 2006. 356 p.

SAHNI, S**. Data Structures, Algorithms, And Applications in C++,** 2nd Edition. Hyderabad, India. Silicon Press, 2005. 792 p.

THAREJA, R. **Data Structures Using C,** 2nd Edition. New Delhi, India. Oxford University Press, 2014. 504 p.