Relatório Estrutura de Dados

Estática

Este relatório apresenta a implementação de uma lista utilizando alocação estática em linguagem C. O objetivo deste trabalho é implementar as operações básicas de inserção no início, fim e em uma posição específica, além de imprimir a lista. Adicionalmente, foram realizados testes de desempenho para comparar o tempo de execução das operações de inserção em diferentes posições.

A lista é definida como uma estrutura que contém um array de inteiros e um inteiro que representa o número de elementos na lista. A implementação incluiu as funções IsFull, IsEmpty, insertAtStart, `insertAtEnd`, `insertAtPosition` e `printList.`

A função `insertAtStart` insere um elemento no início da lista, deslocando os elementos já existentes para a direita. A função `insertAtEnd` insere um elemento no fim da lista. Já a função i`nsertAtPosition` insere um elemento em uma posição específica, deslocando os elementos subsequentes para a direita.

Para avaliar o desempenho das funções de inserção, foram realizados testes em que foram gerados 10.000 números inteiros aleatórios e inseridos na lista em diferentes posições. A operação de inserção no início foi realizada para um terço dos números, a inserção em uma posição aleatória para outro terço dos números, e a inserção no fim para o último terço dos números. O tempo de execução das operações foi medido utilizando a função clock() da biblioteca `time.h.`

Os testes de desempenho mostraram que a inserção no início da lista é a operação mais custosa, seguida da inserção em uma posição específica e, por último, a inserção no fim da lista. Em média, a inserção no início levou cerca de 0,005 segundos, enquanto a inserção em uma posição específica levou cerca de 0,003 segundos e a inserção no fim levou cerca de 0,002 segundos.

A implementação de uma lista com alocação estática em linguagem C foi realizada com sucesso, e as operações básicas de inserção no início, fim e em uma posição específica, bem como a impressão da lista, foram implementadas. Além disso, os testes de desempenho mostraram que a inserção no início é a operação mais custosa em termos de tempo de execução.

Encadeada

O objetivo deste relatório é apresentar a implementação de uma lista encadeada com alocação dinâmica em linguagem C, utilizando um modelo de nó com um inteiro e um ponteiro para outro nó. A lista foi implementada com funções para inserir elementos no início, no fim e em uma determinada posição, além de uma função para imprimir a lista.

A implementação foi dividida em duas estruturas, uma para o nó e outra para a lista encadeada. A estrutura do nó possui dois campos: um inteiro para armazenar o valor e um ponteiro para o próximo nó. A estrutura da lista encadeada contém apenas dois ponteiros, um para o primeiro nó (cabeça) e outro para o último nó (cauda).

A função `CreateList` é responsável por criar uma lista vazia e retorna um ponteiro para a lista criada. Ela aloca dinamicamente espaço na memória para a lista e inicializa a cabeça e a cauda com NULL.

A função `InsertAtStart` insere um elemento no início da lista. Ela aloca dinamicamente espaço na memória para o novo nó, define o valor passado como parâmetro e o ponteiro para o próximo nó como NULL. Se a lista estiver vazia, o novo nó é definido como a cabeça e a cauda, caso contrário, o novo nó é definido como a nova cabeça da lista.

A função `InsertAtEnd` insere um elemento no final da lista. Ela aloca dinamicamente espaço na memória para o novo nó, define o valor passado como parâmetro e o ponteiro para o próximo nó como NULL. Se a lista estiver vazia, o novo nó é definido como a cabeça e a cauda, caso contrário, o novo nó é adicionado ao final da lista é definido como a nova cauda.

A função `InsertAtPosition` insere um elemento em uma posição específica na lista. Ela aloca dinamicamente espaço na memória para o novo nó, define o valor passado como parâmetro e o ponteiro para o próximo nó como NULL. Se a lista estiver vazia, o novo nó é definido como a cabeça e a cauda. Caso contrário, é percorrida a lista até a posição desejada e o novo nó é adicionado entre os nós da posição anterior e posterior.

A função `printList` imprime a lista encadeada na tela. Se a lista estiver vazia, imprime "Lista vazia :(". Caso contrário, percorre a lista até o final e imprime cada valor.

Para testar a implementação, foram criadas as funções `InsertAtStartTest`, InsertAtEndTest e `InsertAtPositionTest`, que inserem 10.000 elementos em suas respectivas posições na lista e medem o tempo de execução de cada função com a função clock da biblioteca time.h.

A lista encadeada é uma estrutura de dados fundamental na programação e pode ser utilizada em diversos contextos para armazenar e manipular dados de forma dinâmica. A implementação apresentada neste relatório permite inserir elementos em qualquer posição da lista de forma.

Duplamente Encadeada

Este código é uma implementação de uma lista duplamente encadeada com alocação dinâmica, feita em C. A lista permite a inserção de elementos no início, final e em qualquer posição da lista.

O código começa com a definição de uma estrutura de nó, que contém um inteiro, um ponteiro para o próximo nó e um ponteiro para o nó anterior. Em seguida, é definida uma estrutura de lista encadeada que contém um ponteiro para o primeiro nó da lista (head) e um ponteiro para o último nó da lista (rear).

O código define seis funções, sendo elas:

- `CreateList()`: cria uma nova lista e retorna um ponteiro para ela.
- `InsertAtStart()`: insere um nó no início da lista.
- I`nsertAtEnd()`: insere um nó no final da lista.
- `InsertAtPosition()`: insere um nó em uma posição específica na lista.
- `PrintLinkedList()`: imprime a lista encadeada.
- `main()`: a função principal do programa.

A função `CreateList()` aloca espaço para uma nova lista, inicializa os ponteiros da cabeça e cauda para NULL e retorna um ponteiro para a nova lista.

A função `InsertAtStart()` aloca espaço para um novo nó, define o seu próximo e anterior como NULL e, se a lista estiver vazia, define o novo nó como a cabeça e a cauda da lista. Caso contrário, o novo nó é definido como o novo primeiro nó da lista e o seu próximo aponta para o nó antigo.

A função I`nsertAtEnd()` aloca espaço para um novo nó, define o seu próximo e anterior como NULL e, se a lista estiver vazia, define o novo nó como a cabeça e a cauda da lista. Caso contrário, o novo nó é definido como o último nó da lista, e o nó anterior da cauda é atualizado para apontar para o novo nó.

A função `InsertAtPosition()` aloca espaço para um novo nó, define o seu próximo e anterior como `NULL` e, se a lista estiver vazia, define o novo nó como a cabeça e a cauda da lista. Caso contrário, a função percorre a lista até encontrar a posição especificada, ou o último nó, e insere o novo nó após o nó anterior.

A função `PrintLinkedList()` imprime a lista encadeada, percorrendo a lista a partir da cabeça e imprimindo o valor do dado armazenado em cada nó.

Por fim, a função principal do programa, main(), cria uma nova lista, insere 1/3 das entradas no início, 1/3 no fim e 1/3 em posições aleatórias. Em seguida, imprime a lista encadeada. O programa também mede o tempo de inserção no início da lista, no fim e em posições aleatórias, utilizando a função `clock()` da biblioteca `time.h`.

Testes Realizados:

Para testar a eficiência das listas implementadas, foram realizados três testes para cada lista: inserção no início, inserção no final e inserção em posição aleatória. Em cada um dos testes, foram gerados 10.000 números aleatórios com valores entre 0 e 999 e inseridos nas listas. Para cada tipo de inserção, um terço dos números foram inseridos nas listas tendo como resultados os dados da tabela abaixo:

**Para 10 testes com inserção de 10.000 dados*

Estrutura	Qtd. Ins. Início	Qtd. Ins. Fim	Qtd. Ins. Posição	Menor tempo	Maior Tempo	Tempo Médio
Lista estática com capacidade de 10.000	3333	3333	3333	0,028	0,113	0,0436
Lista dinâmica com encadeamento (simples)	3333	3333	3333	0,051	0,065	0,0587
Lista dinâmica duplamente encadeada	3333	3333	3333	0,057	0,069	0,0624

UFMT - Ciência da Computação
Disciplina de ED 1

Prof. Ivairton

Wesley Antonio Junior Dos santos
RGA: 202011722024

Felipe Rodrigues Dantas
RGA: 202111722010

2023-03-15