Hérick Vitor Vieira Bittencourt

UNIVALI - Universidade do Vale de Itajaí

Ciências da Computação – Estrutura de Dados

**Trabalho M1, Parte 2 – Manipulador de Polinômios**

Professor – Marcos Cesar Cardoso Carrard

Data *-* 13/04/2023

[herick@edu.univali.br](mailto:herick@edu.univali.br)

**Descrição da estrutura e Análise de eficiência**

Começando pela estrutura, o programa é composto por três arquivos main.cpp, cujo contém a função main, além de variáveis e funções relacionadas ao menu, no segundo arquivo, parte1.h, armazena todas a estrutura e operações relacionadas aos polinômios, já o terceiro arquivo, interface.h, armazena funções adicionais para facilitar a implementação do menu com a biblioteca da parte 1, contendo funções para leitura correta de dados e operações relacionadas ao terminal, como limparTela e esperarInput.  
Para uma estrutura mais compreensível e fácil de implementar, o menu utiliza recursividade, chamando a si mesmo por padrão com o ID 0, tendo este valor substituído conforme é navegado, a lógica para manter cada identificação de submenu relativamente afastada uma da outra é que para cada submenu, seus Ids internos são multiplicados por um valor na base 10, assim é possível manter todo o menu em um switch case e apenas retornar para o começo do loop caso o programa execute uma operação ou não encontre um case valido, facilitando a leitura e impedindo Stack overflow.

Texto

Descrição gerada automaticamente

Passando para a parte1.h, o sistema de polinômios utiliza as listas duplamente encadeadas para seu funcionamento devido a operação deletarExpoente utilizar os elos anteriores para reajustar a lista ao remover um monômio localizado no final.

Texto

Descrição gerada automaticamente

Os polinômios possuem a mesma lógica de inicialização das listas duplamente encadeadas, contendo apenas como diferença o armazenamento de dois valores, a constante e o expoente, anteriormente havia um booleano para representar valores apenas constantes, no entanto todas as suas referências foram removidas por permitir a criação de não-monômios em um polinômio.

Texto

Descrição gerada automaticamente

A função para mostrar o polinômio (anteriormente conhecida como LerPolinomio, agora MostrarPolinomio) utiliza um while loop para passar por cada monômio, seguido do display do valor em formato legível utilizando uma sequência de condições ternárias para avaliar o monômio, a operação foi feita desta forma devido a quantidade menor de instruções resultantes de uma sequência para uma única saída do que se fosse uma sequência de ifs statements e couts, aumentando assim a sua eficiência.  
Sequência de avaliações via condição ternária:

1. Display do símbolo **+** para valores positivos.
2. Display do símbolo – para o caso em que a constante seja -1X, já que nesta situação o -1 não é renderizado na condição 3.
3. Display da constante caso seja elevado a zero, ou caso possua um expoente, é apenas renderizado se o valor é maior que 1 ou menor que -1, impedindo de aparecer 1X e -1X.
4. Display de X caso o expoente seja diferente de zero.
5. Display do expoente caso seja maior que 1 ou menor que -1, além de a constante ser diferente de zero.
6. Espaçamento caso o monômio tenha sido renderizado (caso a constante seja zero, nenhuma das condições é verdadeira e apenas resulta em uma string vazia).

Texto

Descrição gerada automaticamente

A função acharExpoente utiliza um while loop até retornar um No caso o polinômio possua o expoente desejado, caso não haja nenhum No com o expoente alvo, a função retorna NULL após percorrer toda a lista, devido a busca sequencial de listas encadeadas, não foi possível encontrar possíveis melhorias nesta função.

Tela preta com letras brancas

Descrição gerada automaticamente

A função inserirMonomio começa criando um monômio seguindo os parâmetros fornecidos ou retornando caso não haja uma lista para inserir ou a constante seja zero, antes de tentar inserir na lista conforme a ordem, primeiro é chamado acharExpoente para encontrar uma possível soma de monômios com expoentes similares, caso encontre, as constantes são somadas e a função é completada, caso contrário, a função verifica se a lista está vazia, ou se o novo monômio é superior ao começo do polinômio/menor que o final do polinômio e insere-o em seu lugar respectivo, caso ainda não tenha encontrado onde inserir, um while loop é utilizado para encontrar o ponto em que se o monômio ficar, seu vizinho da esquerda possuirá um expoente maior e seu vizinho da direita possuirá um expoente menor e então insere-o e a função é concluída, a eficiência desta operação é relativa ao estado e tamanho da lista fornecida, no melhor cenário, a constante será zero ou ocorrerá uma soma com o maior valor (o primeiro da lista), no pior cenário, é um novo monômio cujo expoente não pode ficar nem no começo, nem no fim da lista, ficando no penúltimo ponto possível, resultando em quase duas pesquisas sequenciais completas pela lista.  
Texto

Descrição gerada automaticamente

inserirMonomioFinal é uma variação simples de inserirMonomio, ignorando todas as regras de inserção e diretamente inserindo no final da lista o novo monômio.

Texto

Descrição gerada automaticamente

deletarExpoente utiliza acharExpoente para encontrar um No com o mesmo expoente, caso não encontre, a função termina em falha por não encontrar o alvo, caso encontre, seus elos vizinhos são verificados para certificar de que está no meio, caso esteja, os elos vizinhos se conectam, caso contrário, o monômio define outro para ser o começo/fim dependendo se sua posição, ao final dos ajustes para manter a estrutura da lista, o No alvo é deletado da memória, novamente, sua eficiência é relativa ao tamanho da lista e seu estado, no melhor caso, o alvo é o segundo elemento da lista, pois ativará a condição principal, no pior caso, o alvo para deletar estará no final da lista, necessitando caminhar por toda a lista e então pular para a última possível condição da função, resultado NULL não é a melhor condição pois ainda necessitará que acharExpoente caminhe por toda a lista para chegar ao resultado, e não sendo a pior pois o retorno é na linha seguinte.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

A função somarPolinomio utiliza dois polinômios como argumento, ela cria e inicializa uma nova lista, em seguida dois while loops são feitos e os valores de ambas as listas são inseridos na nova lista, retornando ela mesma após a conclusão, a função subtrairPolinomios utiliza a mesma lógica, no entanto, as constantes da segunda lista tem seu sinal invertido ao ser adicionado na nova lista, resultando assim na subtração entre os dois polinômios, a eficiência das funções é relativa ao tamanho das listas, no melhor cenário, não há nada o que somar nas duas listas, retornando apenas uma terceira lista vazia, no pior cenário, as duas listas possuem muitos elementos e na subtração nenhum dos monômios se anulam.

Texto

Descrição gerada automaticamente

A função multiplicacaoEscalar requer uma lista e um valor inteiro como argumentos, ao iniciar, uma nova lista é criada, então, usando um while loop, todos os valores da lista original são inseridos e suas constantes são multiplicadas pelo valor escalar, ao final da operação, a nova lista é retornada, a eficiência da função é relativa ao tamanho da lista, uma lista vazia sendo o melhor cenário, e no pior cenário, uma lista enorme.  
Texto

Descrição gerada automaticamente

A função multiplicarPolinomios utiliza duas listas como argumentos, uma nova lista é criada e utilizando dois while loops, com o segundo sendo relacionado a segunda lista, dentro do primeiro loop, são inseridos monômios na nova lista, cujo são resultados da propriedade distributiva entre os dois polinômios, suas constantes são multiplicadas e expoentes somados, retornando então a lista resultante da operação, a eficiência desta função é relativa ao tamanho de ambas as listas, com prioridade na primeira lista, no melhor cenário, a primeira lista não possui nada e o resultado é uma lista vazia, no pior cenário, as duas listas são grandes.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

A ultima operação da biblioteca é determinarValor, cujo requer uma lista e um valor para X como parâmetro, ela inicia criando uma variável inteira para armazenar o resultado da operação, seguido então de um while loop pela lista, caso o expoente seja diferente de zero, o resultado é somado pela multiplicação da constante com a potência do valor de X e o expoente do monômio, caso contrário, a constante é apenas somada a variável resultado, já que qualquer valor de X se for elevado a zero resultará em 1, ao final do loop, é retornado o valor numérico do polinômio quando o valor de X for o valor fornecido.

Texto

Descrição gerada automaticamente

**Aplicação final**

A conclusão do trabalho resultou em uma aplicação responsável pela manipulação de polinômios, dentro dela, é possível criar novos polinômios, inserir monômios, encontrar e remover monômios, além de realizar operações de/entre os polinômios como soma, subtração, multiplicação e determinação de valores, todas estas operações cujo resultam na criação de um polinômio são armazenadas em um array de 50 slots, por fim, o menu possui embutido a partir de uma função da biblioteca da parte 1 uma função que realiza uma série de operações para demonstrar a aplicação em funcionamento com todas as funções.

Texto

Descrição gerada automaticamente

Figura : Exemplo da multiplicação de polinômios (p1 \* p2 = p3)

Texto

Descrição gerada automaticamente

Figura 2: demonstração do menu principal

Tela de computador com jogo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Figuras 3 e 4: Demonstração da recursividade via breakpoint (fig.3 é durante a operação 1-1, fig.4 é após voltar ao menu, Stack resetou)

Texto

Descrição gerada automaticamente

Figura 5: Trecho do código de exemplo