Relatório de Desenvolvimento da Lista de Exercício 1

Bruno Da Fonseca Chevitarese (20231BSI0082)

Marcos Vinícius Souza dos Santos (20222BSI0156)

Outubro 2024

Sumário

1 Apresentação

Este documento tem por objetivo contemplar a primeira lista de exercícios propostos na Disciplina Linguanges Formais e Autômatos, lecionada pelo professor Doutor Jefferson de Oliveira Andrade, ofertado no Instituto de Ensino, Ciências e Tecnologia do Espírito Santo (Ifes), *campus* Serra. As seções seguintes contém as respostas para os problemas propostos de 1 a 4.

Cabe destacar que todos os problemas possuem o mesmo padrão de diretórios e projetos. Portanto, os *scripts* da solução do problema "x"encontra-se no caminho "/problemas/prblx/src/main/scala/". Ademais, os códigos podem estar fragmentados em múltiplos arquivos por questões de organização e legibilidade. Por fim, as funções principais permite acesso a todas as funcionalidades necessárias para resolução do problema, mas não executam em *loop* infinito.

2 O Problema das Bases Numéricas

2.1 Enunciado

Escreva um programa que converte um número inteiro de uma base numérica para outra. O programa deve receber como entrada o número a ser convertido, a base original e a base para a qual ele deve ser convertido. As bases suportadas devem ser 2 (binária), 8 (octal), 10 (decimal), 12 (duodecimal), 16 (hexadecimal), e 20 (vigesimal).

2.2 Solução Proposta

O problema 1 consiste na conversão entre bases numéricas. Como sabe-se, é possível converter qualquer número de uma dada base numérica para outra base numérica, basta saber quais as suas regras de conversão. O problema originado desta abordagem é a criação de uma função específica para cada conversão. Assim sendo, optou-se pela abordagem de converter um número para uma base específica e a partir dessa base realizar a conversão para a base desejada.

A base numérica central escolhida foi a base decimal, dada a simplicidade da implementação dos algoritmos de conversão. Para converter qualquer número da base decimal para uma base genérica, basta ler os restos da divisão modular do número na base decimal pelo número de caracteres diferentes da outra base de forma inversa à realização das operações. Já a conversão de qualquer outra base para a base decimal se dá pelo produtório do número de caracteres diferentes daquela base seguido da soma pelo valor atual identificado.

Para a legibilidade do código e do projeto, foram gerados vários *scripts*, cada um com as respectivas funções de conversão. Após finalizar a elaboração dos *scripts*, percebi o quão pateta eu fui. Dentro da pasta "convert_to", que contém todos os *scripts* de conversão, criou-se um *script "Generic"*, que contém funções que convertem qualquer número de uma base genérica para base decimal e qualquer número na base decimal para a base numérica. Qualquer uma das duas soluções é válida, mas uma é mais esperta que a outra.

3 O Problema dos Parênteses Mal Formatados

3.1 Enunciado

Desenvolva um programa que verifica se uma expressão composta apenas de parênteses (‘(’ e ‘)’) está corretamente balanceada, ou seja, se cada parêntese de abertura tem um correspondente parêntese de fechamento. O programa receberá como entrada uma string contendo apenas parênteses, e como saída uma mensagem que indica se a expressão está bem formada.

3.2 Solução Proposta

O Problema 2 consiste na determinação se uma *string* de entrada possui parênteses balanceados. Como foi explicado que a string de entrada iria conter somente os caracteres "("e ")"não foi realizado nenhum tipo de tratamento de erros.

Considerando que cada abertura de parênteses exige um correspondente de fechamento, pode-se resolver esse problema através de um simples algoritmo. Dada a *string* de entrada, basta avaliar caractere a caractere e verificar se ele é do tipo "("ou ")". Caso o caractere seja do tipo "(", deve-se adicioná-lo a uma pilha. Caso o caractere seja do tipo ")"deve-se verificar, primeiro, se a pilha está vazia e depois remover o último caractere inserido. Por fim, ao final da execução, basta verificar se a pilha ainda possui caracteres.

Esse algoritmo é muito simples e funciona graças ao funcionamento de uma pilha. Por definição, uma pilha é uma estrutura de dado onde o último elemento inserido é o próximo a sair. Assim, ao iniciar o código, todos os caracteres "("serão inseridos e para cada ")"um será removido. Caso restem caracteres na pilha, há mais caracteres "("do quê caracteres ")". Além disso, quando não houverem caracteres "("na pilha, entende-se que ela está vazia. Ao cada encontro de um caractere ")"na pilha, caso ela esteja vazia, entende-se que há mais caracteres ")"do que "(". Por fim, caso no final da execução a pilha não esteja vazia, não há fechamento de cada abertura de parênteses.

4 O Problema dos Números Perfeitos

4.1 Enunciado

Escreva um programa que verifique se um número é perfeito. Um número perfeito é aquele que é igual à soma de seus divisores próprios, i.e., os divisores excluindo o um o próprio número. O programa receberá como entrada um número inteiro positivo, e como saída uma mensagem que indica se o número é perfeito ou não.

4.2 Correção do Enunciado

Antes de inciar a resolução do problema, deve-se verificar que há um pequeno equívoco no enunciado. Um número perfeito é, por definição, um número cuja a soma de seus divisores incluso o 1 e excluso ele próprio é igual a ele próprio; ou mesmo um número perfeito é tal qual a soma de seus divisores é igual ao dobro do próprio número. A partir dessa nova definição, segue-se, na seção seguinte, a solução proposta.

4.3 Solução Proposta

O algoritmo de resolução do problema 3 é relativamente simples. Basta pegar os divisores de um número, incluso o 1 e excluso ele próprio, somá-los e verificar se a soma é igual ao número original.

Dentro do contexto de divisão modular, um número x é divisível por y quando o resto da divisão de x por y é côngruo a 0. Ou seja: $x \equiv 0 \ (mod \ y)$. Isto posto, pode-se desenvolver um algoritmo recursivo que fornece o resultado. Os parâmetros de entrada da função são: o número que busca-se dividir, o valor do divisor atualmente sendo testado e um outro parâmetro acumulador. Caso o resto da divisão do parâmetro "numero"pelo parâmetro "divisor"seja 0, realiza-se uma chamada recursiva à função com o divisor acrescido 1 e o acumulador somado ao divisor. Caso o "divisor"seja 1, realiza-se uma chamada recursiva à função com o "divisor"acrescido 1 e os demais parâmetros iguais. Por fim, a condição de parada é o parâmetro divisor ultrapassar a metade do número fornecido (ou seja, não haverem mais divisores para aquele número).

5 O Problema do Cálculo com Álgebra Relacional

Não Implementado.