Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada
Linguagens de Programação: Conceitos e Paradigmas

Problemas para Experimentação dos Interpretadores

Este documento descreve os problemas que serão utilizados para a avaliação dos interpretadores do projeto da disciplina Linguagem de Programação: Conceitos e Paradigmas. Para cada problema, crie um programa que o solucione utilizando a linguagem proposta pelo grupo, ou seja, aquela aceita pelo interpretador criado pelo grupo. Esses programas, *ou versões com pequenas modificações*, serão executados sobre um conjunto de dados de entrada durante a avaliação. As soluções apresentadas devem ser adaptadas às características de cada linguagem, mas sempre atendendo às funcionalidades exigidas.

Problema 1:

Crie um programa que, dados três valores numéricos x, y e c, onde x e y são números racionais e c é um número inteiro, previamente armazenados no código-fonte, avalia a expressão $x^2 - y + c$ e imprime seu resultado na tela.

Problema 2:

Crie um programa que leia uma quantidade desconhecida de números e informe quantos deles estão nos seguintes intervalos fechados: [0, 25], [26, 50], [51, 75] e [76, 100]. A entrada de dados deve terminar quando for lido um número negativo.

Problema 3:

Crie um programa que leia duas matrizes numéricas e, quando possível, imprima a soma e o produto dessas matrizes. Caso uma operação não possa ser realizada para as matrizes lidas, imprima uma mensagem informando da impossibilidade.

Problema 4:

Defina o tipo $rational\_t$ para representar números racionais. O tipo $rational\_t$ deve ser representado como um registro (ou tipo correspondente) com campos inteiros *numerador* e *denominador*. Em seguida, escreva os seguintes subprogramas:

A) Subprograma que, dados dois parâmetros inteiros a e b, onde $b \neq 0$, retorna um valor $rational\_t$ para representar a fração a/b.

B) Subprograma que, dados dois parâmetros do tipo $rational\_t$, retorna $true$ se eles representam o mesmo número racional ou $false$, em caso contrário.

C) Subprogramas que retornem um valor $rational\_t$ correspondente a soma, negação, subtração, multiplicação, inverso e divisão entre valores $rational\_t$, passados como parâmetros (um subprograma por operação).

No programa principal, invoque cada um dos subprogramas e imprima os resultados produzidos, indicando numerador e denominador.

Problema 5:

Crie um subprograma chamado *mdc*, com três argumentos n, m (passados por valor) e r (passado por referência), nesta ordem. O subprograma *mdc* deve calcular o maior divisor comum entre dois números naturais estritamente positivos n e m, de acordo com o seguinte algoritmo recursivo:

- Se n for um divisor de m, n é o maior divisor comum de n e m.
- Se m for um divisor de n, m é o maior divisor comum de n e m.
- Se n não for um divisor de m, e se m for maior que n, então o maior divisor comum de m e n é também o maior divisor comum de n e do resto da divisão de m por n.

O subprograma deve retornar seu resultado por meio de parâmetro r, que deve ser posteriormente impresso na tela pelo programa principal.

Problema 6[1]:

Uma árvore binária de busca generaliza a ideia de listas encadeadas crescentes. Em uma árvore binária de busca, os nós têm um campo chave de um tipo ordenável e apresentam as seguintes propriedades: para qualquer nó n, a chave de n é maior ou igual à chave de qualquer nó na subárvore esquerda de n e menor ou igual à chave de qualquer nó na subárvore direita de n. Implemente uma árvore binária de busca com chaves de tipo inteiro e as seguintes operações:

A) Transforme uma sequência de valores em uma árvore binária de busca.

B) Encontre a chave mínima da árvore, indicando seu nível.

C) Encontre a chave máxima da árvore, indicando seu nível.

D) Imprima a árvore de busca na saída padrão, nível a nível.

[1]Texto adaptado de https://www.ime.usp.br/~pf/algoritmos/aulas/binst.html