# SCC0502 Algoritmos e Estruturas de Dados 1

## Exercício 4 - Números Astronômicos

### Objetivo

O objetivo deste exercício prático é estimular os estudantes a se familiarizarem com o comportamento e a lógica associados à estrutura de dados **lista encadeada.**

Queremos que os alunos se habituem com a implementação desssa estrutura e consigam entender as funções básicas, que garantem a execução adequada desse *TAD*, em diferentes contextos.

Para acostumar os alunos com conceitos de modularização e boas práticas de escrita de código, será exigido o uso de múltiplos arquivos *.c* e *.h* no projeto, bem como a construção de um arquivo **Makefile**, responsável por gerenciar a execução do programa.

### Descrição

Acredito que em outras disciplinas aprenderam que, comumente, valores inteiros - *int* - são armazenados com **4 bytes** de memória, e valores decimais - *double* - são armazenados com **8 bytes**.

Devido a essa estruturação, existe um limite físico sobre o valor que podemos registrar com essas variáveis.

- 4294967295 para um inteiro sem sinal atribuído **UINT_MAX**;
- 3.402823e+38 para o tipo ponto flutuante **FLT_MAX**;

Referência:

https://docs.microsoft.com/en-us/cpp/c-language/cpp-integer-limits?view=msvc-160

https://en.cppreference.com/w/c/types/limits

Acreditam que esse limite é suficiente para representar todos os fenômenos da realidade?

Existem diversos contextos, que atravessão campos desde a estatísca até a cosmologia que precisam conseguir lidar com números que possuem uma ordem de magnitude muito superior aos limites apresentados acima.

- O maior número primo calculado até então possuí **24,862,048** dígitos;
- O modelo do *Big Bang*, por exemplo, sugere que o universo tem **13.8 bilhões** de anos - 4.355e+17 segundos-
- O universo observável está a 93 bilhões de anos luz, aproximadamente 8,7984793395e+23 km;

Existe uma maneira de representar esses números gigantescos por meio de uma lista encadeada. Pode-se dividir os digitos dos algarismos entre os nós da lista, caso queiramos represntar os 93 bilhões de anos poderíamos fazer:

```
    NULL  <-- (930)  <-- (0000)  <--  (0000)
```

O objetivo deste exercício é desenvolver um projeto que seja capaz de realizar operações aritméticas e lógicas sobre esses números.

### Entrada

Seu algoritmo deve receber como entrada um número *n*, que representa a quantidade de operações que serão requisitadas. Após isso ele receberá um sequência de n comandos para serem executados dentre os possíveis *{soma, maior, menor, igual}* acompanhados de dois números inteiros sobre os quais serão realizadas as operações;

```
12
soma 9 3
soma 225 225
soma 11123456789 11987654321
soma 101498473623545 10234586723
soma 1123456 1123459
maior -10 1
menor 012143 110
maior 1123456 -112345664
igual 123456789745 123456789745
soma 050 050
soma 2500 113567
igual 09870 098700
```

### Saída

Como saída, seu programa deve indicar o resultado das operações

```
Resultado :: 12
Resultado :: 450
Resultado :: 23111111110
Resultado :: 101508708210268
Resultado :: 2246915
Resultado :: False
Resultado :: False
Resultado :: True
Resultado :: True
Resultado :: 100
Resultado :: 116067
Resultado :: False
```

## Observações da implementação

Como é descrito na sessão objetivos desse documento, não queremos apenas que os alunos resolvam o problema, mas que utilizem métodos que serão comuns no decorrer da disciplina.

Devido a esse objetivo, será exigido que vocês desenvolvam seu projeto representando o objeto **big number** como struct e com a implementação do **TAD lista** para construir as operações de forma abstrata. A memória deve ser alocada **dinamicamente** e ser devidamente liberada ao fim da execução, ou seja, usem a estrutura **lista encadeada dinâmica**, ligação simples (sentido único).

Para cada nó da lista, como foi desenhado acima, representem o valor dentro com nó com o tipo **int**.

Utilizar múltiplos arquivos *.c* e *.h* para separar a implementação e a responsabilidade dos métodos de cada objeto.

Construir funções para realizar operações repetitivas, ou seja, modularizar adequadamente seu código.

Escrever um arquivo **Makefile** que será responsável por gerenciar a execução do projeto dentro da plataforma *run.codes*.

## Observações da avaliação

A avaliação do seu programa será feita além do resultado da plataforma *run.codes*. Portanto, ter um bom resultado com os casos de teste, não será suficiente para garantir a **nota máxima** e nem a **aprovação do exercício**.

Caso seu projeto não satisfaça os pontos exigidos nos **objetivos** e explicitados nas **observações de implementação**, sua nota poderá ser reduzida ou ser desconsiderada.

Cópias de código entre alunos, acusadas pela plataforma, resultarão imediatamente em **zero** aos dois ou mais alunos envolvidos.