Trabalho 2

Números Astronômicos

Estruturas de Dados I

ICMC-USP

2025

1 Enunciado

Suposição: valores inteiros (`int`) são armazenados com 4 bytes de memória, e valores reais (`double`) são armazenados com 8 bytes. Devido a essa estruturação, existe um limite físico sobre o valor que podemos registrar com essas variáveis. O limite para um inteiro sem sinal é de 4.294.967.295, e para o tipo ponto flutuante é de 3.40282338.

Esse limite, no entanto, não é suficiente para representar todos os fenômenos da realidade. Existem diversos contextos que precisam lidar com números que possuem uma ordem de magnitude muito superior aos limites apresentados acima.

- O maior número primo calculado até então possui 24.862.048 dígitos.
- O modelo do Big Bang sugere que o universo tem 13,8 bilhões de anos ou 4.355×10^{17} segundos.
- O universo observável está a 93 bilhões de anos-luz, aproximadamente $8.798479339523 \times 10^{20}$ km.

Existe uma maneira de representar esses números gigantescos por meio de uma lista encadeada. Pode-se dividir os dígitos dos algarismos entre os nós da lista. Para representar os 93 bilhões, por exemplo, poderíamos fazer:

$$\text{NULL} \leftarrow (930) \leftarrow (0000) \leftarrow (0000)$$

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

2 Especificações do trabalho

O programa deverá ser capaz de realizar operações aritméticas e comparações entre números inteiros de tamanho arbitrário, isto é, números que não podem ser representados por tipos primitivos (`int`, `long int`, etc.).

Esses números deverão ser representados utilizando uma **lista encadeada**, na qual cada nó da lista armazenará um bloco de dígitos do número. Dessa forma, o programa deve permitir somar e comparar números de qualquer comprimento, desde que caibam na memória disponível.

Requisitos principais

1. Criar uma estrutura de dados do tipo `Lista`, na qual cada nó armazena:
   - Um bloco de até k dígitos (por exemplo, 4 dígitos);
   - Um ponteiro para o próximo nó da lista.
2. Implementar funções para:
   - Ler números de tamanho arbitrário a partir da entrada padrão e convertê-los em listas encadeadas.
   - Realizar a **soma** entre dois números grandes representados por listas.
   - Comparar dois números grandes, retornando se o primeiro é **maior**, **menor** ou **igual** ao segundo.
   - Outras funções para auxiliar na resolução do problema.
3. As operações devem ser implementadas manualmente, **sem utilizar bibliotecas prontas de números grandes**.
4. O programa deve tratar corretamente:
   - Números com zeros à esquerda;
   - Números negativos;
   - Números de diferentes tamanhos.

Relatório

O relatório deverá seguir a estrutura:

1. Introdução: breve explicação do problema e o que foi feito para resolvê-lo.
2. Desenvolvimento: explicar a solução e responder às perguntas abaixo.

3. Conclusão: resumo e fechamento da solução e contribuição de cada membro.

No desenvolvimento, é necessário responder às seguintes perguntas:

- Por que foi escolhida essa estrutura e não outra?
- Como ela resolve o problema?
- Como foi a implementação?
- Quais as suas limitações?
- Quais as dificuldades encontradas ao longo da solução?

Organização e boas práticas

Após o período de submissão, os códigos serão revisados buscando boas práticas no desenvolvimento de programas.

- Cada função deve possuir um nome descritivo e ser responsável por uma única tarefa bem definida.
- Documentem o código e deixem comentários pertinentes para seu entendimento, principalmente no começo de funções e loops com lógica importante.
- Desaloquem toda a memória alocada dinamicamente ao fim do código. *Dica: usem o valgrind.*
- Sempre deve-se checar por valores nulos no começo de funções com ponteiros e após tentar realizar uma alocação de memória.

3 Entrada e Saída

Dado um par a e b de números inteiros MUITO grandes, realize as seguintes operações:

- **soma**: retorna $a + b$
- **igual**: se $a == b$, retorna $True$, caso contrário, retorna $False$
- **maior**: se $a > b$, retorna $True$, caso contrário, retorna $False$
- **menor**: se $a < b$, retorna $True$, caso contrário, retorna $False$

Entrada

A primeira linha contém o inteiro n, que representa a quantidade de operações que serão requisitadas. Após isso, ele receberá uma sequência de n comandos para serem executados, que podem ser: `soma`, `maior`, `menor` ou `igual`, acompanhados de dois números inteiros sobre os quais serão realizadas as operações.

Saída

A saída consiste em uma linha, o resultado da operação solicitada. O formato deve seguir essa especificação:

```
Resultado :: <valor_ou_booleano>
```

Exemplo de Entrada

```
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
```

Exemplo de Saída

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

```
Resultado:: 100
Resultado:: 116067
Resultado:: False
```

4 Critérios de avaliação

- Corretude da execução do código.
- Implementação da estrutura de dados.
- Atendimento às especificações do projeto.
- Legibilidade e documentação interna do código.
- Relatório com explicação da resolução.
- Modularização do código

5 Observações Importantes

- A implementação deverá ser feita em C.
- Deverá ser entregue pelo run.codes o zip com os programas e a entrega do relatório pelo e-disciplinas. O nome e o número USP de cada integrante devem constar em comentário no início do código e no relatório.
- O trabalho foi disponibilizado no dia e deverá ser entregue no dia 22/11/2025, até às 23 horas.

Referências

[1] Pereira, Leonardo, *Listas de Exercícios de Estruturas de Dados I*, ICMC-USP, 2022.