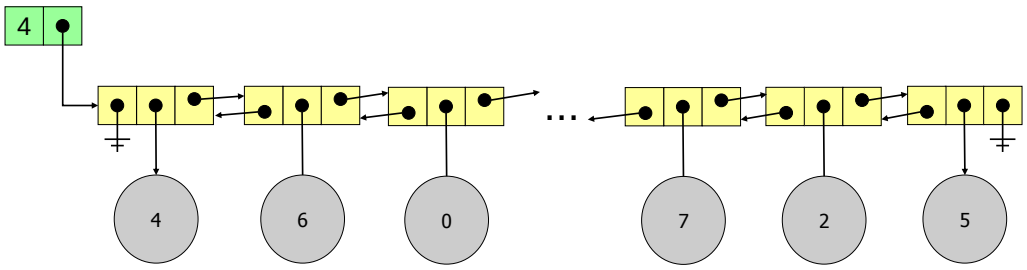
Universidade do Estado de Santa Catarina (UDESC)
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas
Disciplina: Estruturas de Dados I (EDA0001)
Prof. Rui J. Tramontin Jr.

Trabalho: Inteiros Não Negativos com Precisão Infinita

Na área de análise numérica, os tipos de dados oferecidos pelas linguagens de programação são em geral insuficientes para representar números com a precisão necessária. Para números muito grandes pode-se utilizar listas encadeadas para armazenar os dígitos de um número. À medida que o número necessitar de mais dígitos, é possível alocar mais elementos à lista para armazená-los.

O objetivo deste trabalho é implementar uma biblioteca de números inteiros não negativos com precisão infinita (enquanto houver memória disponível). Cada dígito deverá ser armazenado em uma posição da lista. O exemplo a seguir ilustra como o número 4.608.996.124.300.675.543.725 deverá ser representado utilizando uma **lista duplamente encadeada** (LDE):



Para melhorar o desempenho da manipulação da estrutura, o cabeçalho da lista poderá ter um **ponteiro para o último elemento**, tal como na implementação da fila encadeada. Além disso, é possível também armazenar **quantidade de elementos** da lista, ou seja, quantos dígitos o número possui.

O que deve ser feito:

A. Definição e implementação do TDA **`InteiroLongo`**, utilizando uma LDE. Ele deverá implementar as seguintes operações:

1) Inicialização: recebe como parâmetros a referência do TDA e uma cadeia de caracteres (*char* *), e retorna um *int*. O retorno deve indicar erro caso o número representado na cadeia de caracteres não seja válido. Exemplos de uso:

```
a. InteiroLongo x;
b. inicializa_numero(&x, "1234567890987654321"); // Ok
c. inicializa_numero(&x, "123abc9876"); // Erro
d. inicializa_numero(&x, "-109843627"); // Erro
```

2) Desalocação: desaloca todos os dígitos de um número.

3) Mostra número: imprime o número no console, no formato: xxx.xxx.xxx. Por exemplo, o número 12345 deve ser impresso na forma 12.345.

4) Atribuição entre dois números: função do tipo *void* que recebe como parâmetro dois números: o primeiro, por referência, recebe o valor do segundo, por valor.

5) Soma entre dois números: deve ser implementada somando-se dígito a dígito, da direita para a esquerda (fazendo o "vai um" quando necessário).

6) Subtração entre dois números: idem à soma. Deve retornar um código de erro caso o resultado seja negativo.

7) Comparação entre dois números: compara-se dígito a dígito, da esquerda para a direita. Retorna *int*: 0 se forem iguais, +1 se o primeiro for maior, e -1 se o segundo for maior.

8) Multiplicação entre dois números: implementado usando uma sequência de somas. Por exemplo, A x B, significa repetir A vezes e armazenar B em um somatório.

9) Divisão entre dois números: implementado usando uma sequência de subtrações. A dividido por B significa contar quantas vezes A = A - B enquanto A ≥ B. O valor que sobrar em A será o resto da divisão. A função deve calcular tanto o quociente quanto o resto.

10) Potenciação: implementado utilizando uma sequência de multiplicações.

11) Parte Inteira da Raiz Quadrada: a *equação de Pell* permite encontrar a parte inteira de uma raiz quadrada simplesmente subtraindo inteiros ímpares. Por exemplo, para calcular a parte inteira da raiz quadrada de 19, calcula-se a sequência:

Passo 1: 19 – 1 = 18	Como 3 é menor que 9 (o próximo
Passo 2: 18 – 3 = 15	número ímpar), a sequência
Passo 3: 15 – 5 = 10	termina. Como 4 subtrações foram
Passo 4: 10 – 7 = 3	efetuadas, então a resposta é 4.

B. Implementação do **programa principal**: deverá solicitar ao usuário N números que devem ficar armazenados em uma **lista encadeada**. Após a entrada de dados, o programa deve apresentar o seguinte relatório:

1) O maior número;

2) O menor número;

3) A parte inteira da raiz quadrada de todos os números;

4) Mostrar a soma, subtração, multiplicação, divisão (e resto) e potenciação entre pares de números da lista:

- 1º com 2º;
- 2º com 3º;
- ...
- (n-1)-ésimo com n-ésimo.

Observações:

- Na atribuição entre dois números, é preciso ter o cuidado de desalocar os dígitos que já existam, caso a variável que vá receber o valor já tenha sido inicializada previamente;
- Não esqueça de remover os eventuais zeros à esquerda, após a realização de algumas operações;
- Para facilitar a implementação, os valores resultantes das funções devem sempre ser armazenados em uma nova variável. Para exemplificar de maneira conceitual, é muito mais complicado armazenar o resultado de uma soma, por exemplo, na mesma variável:

  A ← A + B // muito complicado de implementar...

  deve ser feito como:

  X ← A + B
  A ← X

Critérios de avaliação:

- **Coletivos**: execução correta e alinhamento com o que foi solicitado neste enunciado. Boas práticas para criação e uso de TDAs também serão avaliadas.
- **Individuais**: domínio do assunto e do programa implementado.

Informações importantes:

- **Equipe**: 2 ou 3 alunos.
- **Entrega**: por e-mail, até o dia **06/12**.
- **Apresentação**: dia **06/12**, no laboratório, durante o horário de aula. Alternativamente, as equipes que preferirem apresentar antes podem agendar um horário para apresentação na sala do professor.