Universidade Federal do ABC
Centro de Matemática, Computação e Cognição
MCTA02 8-15 - Programação Estruturada
Prof. Dr º Maycon Sambinelli
Alunas:
Emily Guidi - RA 11202320595
Paloma Lima - RA 11202021803

READ ME

Santo André – SP
2024/2025

Estrutura e Organização

O código apresentado implementa uma classe para manipular números inteiros arbitrariamente grandes, denominados *BigNumbers*. A estrutura principal é uma lista duplamente encadeada, onde cada nó representa um dígito do número. A nossa estrutura de scripts foi a minima de bignumber.h , bignumber.c , makefile e cliente.c.

Funcionalidades Implementadas

- **Criação e destruição:** As funções createBigNumber e freeBigNumber permitem alocar e liberar memória para os objetos BigNumber.
- **Operações aritméticas básicas:**
    - **Adição:** Implementada de forma eficiente, considerando o carry.
    - **Subtração:** Implementada de forma similar à adição, com o cuidado de lidar com empréstimos.
    - **Multiplicação:** Otimização da multiplicação de números grandes.
    - **Divisão:** Implementada usando um algoritmo de divisão longa.
    - **Exponenciação:** Utiliza o método de exponenciação por quadrados para otimizar o cálculo de potências.
    - **Módulo:** Calcula o resto da divisão.
- **Comparação:** A função compareBigNumbers compara a magnitude de dois números.
- **Conversão:** Converte entre strings e BigNumbers.
- **Cópia:** Copia um número BigNumber integralmente.
- **Gerenciamento de sinal:** A classe BigNumber suporta números negativos.
- **Leitura de entrada dinâmica:** A função readinput permite ler números de tamanho arbitrário da entrada padrão.
- **Alocação dinâmica de memória:** O código utiliza malloc e free para alocar e liberar memória conforme necessário.
- **Algoritmos eficientes:** Exponenciação por quadrados otimiza o desempenho para números grandes, divisão de carry para adição, manipulação de dígitos dos bignumbers para maior rapidez e performance.

Interface Pública

A interface pública da classe BigNumber é composta pelas funções descritas anteriormente. Essas funções permitem criar, manipular e destruir objetos BigNumber, e estão listadas no arquivo bignumber.h.

Lista de funções na interface pública:

BigNumber *createBigNumber(const char *str); void freeBigNumber(BigNumber *bn); BigNumber *addPositiveBigNumbers(BigNumber *a, BigNumber *b); BigNumber *addBigNumbers(BigNumber *a, BigNumber *b); BigNumber *subtractPositiveBigNumbers(BigNumber *a, BigNumber *b) ; BigNumber *subtractBigNumbers(BigNumber *a, BigNumber *b); BigNumber

*createBigNumberFromBigNumber(BigNumber *src); int compareBigNumbers(BigNumber *a, BigNumber *b); void printBigNumber(BigNumber *bn); BigNumber *divideBigNumbers(BigNumber *a, BigNumber *b); int getLength(BigNumber *a); char *readinput(); BigNumber *multiplyBigNumbers(BigNumber *a, BigNumber *b); BigNumber *stringToBigNumber(const char *str); BigNumber *exponenciacao(BigNumber *base, BigNumber *exponent); BigNumber *restoDivisao(BigNumber *a, BigNumber *b);

Algoritmos e Estruturas de Dados Avançadas

- **Exponenciação por quadrados:** Um algoritmo eficiente para calcular potências.
- **Lista duplamente encadeada:** A estrutura de dados principal, escolhida por sua flexibilidade para inserir e remover elementos em qualquer posição.

Divisão de tarefas do trabalho

Emily:

- ❖ Criação da estrutura de arquivos;
- ❖ Função Soma;
- ❖ Função Subtração;
- ❖ Melhoria nos dados de entrada do programa (modelo passado em sala de aula para captar dígito por dígito do Bignumber);
- ❖ Função Multiplicação simples
- ❖ Bateria de testes das três funcionalidades manualmente e com os arquivos de teste;
- ❖ Bateria geral de testes de todas as funções feitas e testes de vazamento de memória;
- ❖ Ajustes em divisão de bignumbers e exponenciação;

Paloma:

- ❖ Elaborar função Divisão (divisão inteira);
- ❖ Elaborar função de exponenciação;
- ❖ Elaborar função de resto de divisão;
- ❖ Testar essas três funções manualmente e com os arquivos de teste;
- ❖ Ajustes em divisão de bignumbers e exponenciação;
- ❖ Escrever o READ ME.