Projeto “bigNumber” de Programação Estruturada

Thiago Perellon Romano 11202130849

O bigNumber desenvolvido utiliza uma struct denominada bigNumber com 3 campos. O primeiro é um vetor de caracteres, feito para armazenar os dígitos do número digitado. O segundo armazena o tamanho do vetor, para possíveis operações que necessitem do tamanho, a fim de evitar o uso excessivo da função strlen(). O terceiro e último campo armazena, em um tipo inteiro, 1 caso o resultado seja negativo e 0 caso seja positivo.

A leitura de cada dígito é feita utilizando a função getchar, que capta todo caractere da entrada do usuário e salva no vetor de caracteres da estrutura bigNumber e, a cada iteração, aumenta a memória alocada para o vetor se necessário. Posteriormente, é chamada uma função que verifica se o primeiro caractere um o sinal de negativo, salva o resultado na variável que representa o valor negativo (isNegative), remove o sinal e move todo o bloco de memória para a esquerda, ocupando da posição 0 em diante do vetor. Após isso, o número é enviado para uma função que vai inverter a posição dos dígitos para facilitar operações posteriores envolvendo soma, subtração e multiplicação.

A interface pública foi simplificada em apenas 1 estrutura de dados e 3 funções principais:

```
typedef struct{
    char *numero; //Salva os digitos do número digitado
    size_t tamanho; //Armazena o tamanho do vetor de números
    int isNegative; //Salva 1 se for negativo e 0 se for positivo
} bigNumber; //Nome da estrutura

bigNumber* readNumber();
bigNumber* verificaOperacao(bigNumber* primeiroNumero, bigNumber* segundoNumero);
void liberaMemoria(bigNumber* numero);
```

A função readNumber não recebe nenhum parâmetro e retorna todos os componentes da estrutura para uma variável criada do tipo bigNumber.

A função verificaOperacao recebe como parâmetro dois bigNumbers. Dentro dessa função é lido o caractere que representa a operação a ser realizada (+, - ou *) e, dependendo do sinal dos números (positivo ou negativo) e do valor absoluto, o tipo de operação e a posição dos números na chamada das funções de operações matemáticas são alterados, facilitando a manipulação dentro das funções que realizam as contas.

Já a função liberaMemoria, é responsável por liberar (free) a memória do vetor de caracteres e, posteriormente, liberar a memória da estrutura, evitando assim vazamento de memória.

Voltando à função verificaOperacao, ela é responsável por gerenciar o tipo de operação a ser executada de acordo com as entradas fornecidas pelo usuário, visando facilitar o processamento das funções. Dentro dela é possível encontrar blocos condicionais para os três tipos de operações estabelecidas no código. Sua principal responsabilidade é: comparar o valor absoluto e os sinais (negativos e positivos) das entradas e, de acordo com os resultados obtidos, manipular a chamada das funções de soma e subtração.

Em casos de soma temos 4 principais casos:

- - A + B
    - Se A > B → - ( A - B )
    - Se B > A → ( B - A )
- A + (- B) = A - B
    - Se A > B → A - B
    - Se B > A → - ( B - A )
- - A + (- B) = - ( A + B )
- A + B, realiza a soma normalmente.

Em casos de subtração, também temos 4 casos principais:

- - A - B = - ( A + B )
- A - (- B) = A + B, realiza a soma normalmente
- - A - (- B) = - A + B
    - Se A > B → - ( A - B )
    - Se B > A → B - A
- A - B
    - Se A > B → A - B
    - Se B > A → - ( B - A )

Em casos de multiplicação, temos apenas dois casos dentro da função verificaOperacao:

- Caso um dos números seja zero, a função de multiplicação não é chamada e o valor 0 é automaticamente atribuído ao vetor resultado
- Caso ambos os números sejam diferentes de zero, realiza a operação normalmente e logo após isso verifica se apenas um dos números de entrada é negativo, em caso positivo, atribui ao campo que carrega o sinal do resultado o valor 1

Finalmente, após a realização das operações, é realizada uma verificação para adicionar o sinal negativo na frente do número resultado caso seja necessário.

Finalmente, Thiago Perellon Romano (11202130849) ficou responsável por 100% do trabalho realizado.

Link do GitHub: [thiagoromano/Projeto-PE (github.com)]

(liberado após 23:59 do dia 10/12/2023)