UNIVERSIDADE FEDERAL DO ABC

PROJETO BIG NUMBERS

Programação Estruturada - Prof. Maycon Sambinelli

Bruno Moraes Filoni 11202130777
Guilherme Carvalho Torres 11202021966

Instruções

A função main recebe como parâmetro um arquivo de entrada .txt para a realização das operações entre Big Numbers e gera um arquivo de saída (.out.txt) com os resultados das operações, como nos exemplos de instâncias fornecidos. Isso possibilita comparar os resultados com o comando diff. Portanto, para executar o arquivo client.c, basta passar como parâmetro o arquivo de entrada desejado. Exemplo: ./client /documents/regular.in.

Representação do Big Number

Em C++, é possível utilizar a classe std::vector para criar um vetor dinâmico, que nada mais é que um array que pode crescer ou diminuir em tamanho durante a execução do programa. Isso possibilita armazenar um Big Number de forma que o limite de tamanho seja a quantidade de memória da máquina que está rodando o código.

Como em C não há vetor dinâmico de forma nativa, foi implementado um struct que simula o vetor dinâmico, que é composto pelo array de ‘signed char’, definido como int8 (int de 8 bits), onde cada elemento representa um dígito do Big number, foi utilizado ‘signed char’, para poupar memória, outro método experimentado foi a concatenação de int’s, onde cada int armazenava 9 dígitos do Big number, poupando ainda mais memória e diminuindo a quantidade de operações necessárias. No entanto, o array de int8 possibilitou armazenar os dígitos direto da stream de input, poupando ainda mais tempo e compensando o maior número de operações. A struct também é composta pela variável nelements, que armazena a quantidade de dígitos do Big number; e a variável signal, que armazena o sinal do Big Number (“-” ou “+” representado por -1 e +1). Este struct pode ser visualizado logo abaixo.

```
typedef signed char int8;
typedef struct {
    int8* data;
    int nelements;
    int signal;
} bignum;
```

Interface pública

```
typedef signed char int8;
typedef struct {int8* data; int nelements; int sign;} bignum;
typedef bignum* BigNumber;

BigNumber bignumber(void);
void bignumber_insert(BigNumber A, int8 a);
void bignumber_free(BigNumber A);

void read_input(BigNumber A);
void read_input_file(BigNumber A, FILE* file);

void reverse(BigNumber A);
void filter_left_zero(BigNumber A);
int max(int a, int b);
int compare(BigNumber A, BigNumber B);

void add(BigNumber A, BigNumber B, BigNumber RES);
void subtract(BigNumber A, BigNumber B, BigNumber RES);
void multiply(BigNumber A, BigNumber B, BigNumber RES);

void bignumber_print(BigNumber A);
void bignumber_print_file(BigNumber A, FILE* out);
int choose_operation(char op, int signal_a, int signal_b);
char* file_name(const char* input);
```

Divisão de Tarefas

Inicialmente a implementação da representação do Big Number como vetor dinâmico ocorreu de forma conjunta. Após esta implementação base, separamos as operações em: soma e subtração implementadas por Guilherme e multiplicação por Bruno. Com as 3 operações já implementadas, Guilherme ficou responsável por organizar a Interface e sua implementação no arquivo bignumber.c, além do makefile. Bruno automatizou os testes e implementou o suporte para ler arquivos de entrada e gerar arquivos de saída.