Universidade Federal do ABC
MCTA028-15 - Programação Estruturada
2018.Q3

Projeto Googol – Parte 2

Professores Emílio Francesquini e Carla Negri Lintzmayer

9 de novembro de 2018

1 Introdução

Nesta segunda fase do projeto vamos estender o código feito para a primeira fase (que já lidava com soma e subtração de números de tamanho arbitrário) para incluir as operações de multiplicação, divisão e resto da divisão.

Assegure-se que as operações de soma e subtração implementadas na primeira fase do projeto funcionam corretamente. Elas são as bases para escrever as operações de multiplicação e divisão.

2 Operações

Será necessário criar uma função para cada operação que será realizada sobre a sua estrutura de dados. Recomendamos (porém a escolha é sua) que você faça funções que alterem o valor passado como parâmetro e não funções que devolvem um novo número com o resultado da operação.

Se você seguiu as sugestões de implementação dadas para a primeira fase do projeto, as seguintes definições serão uma extensão natural às operações já definidas:

```c
/* Multiplica 'a' por 'b' e guarda o resultado (a*b) em 'a'
   'b' não é modificado.
   Devolve o número de dígitos do resultado. */
int multiplica(int a[], int tamA, int b[], int tamB);

/* Divide 'a' por 'b' e guarda o resultado (a/b) em 'a'
   'b' não é modificado.
   Devolve o número de dígitos do resultado. */
int divide(int a[], int tamA, int b[], int tamB);

/* Divide 'a' por 'b' e guarda o resto da divisão (a%b) em 'a'
   'b' não é modificado.
   Devolve o número de dígitos do resultado. */
int resto(int a[], int tamA, int b[], int tamB);
```

Exemplo de uso:

```
/* criando dois números baseados em inteiros: */
int a[TAM_MAX], b[TAM_MAX], c[TAM_MAX], tamA, tamB, tamC;
tamA = criaNumeraoDeInt(4, a);
tamB = criaNumeraoDeInt(3, b);

/* somando os números e imprimindo o resultado, que está no próprio
   'a': */
tamA = soma(a, tamA, b, tamB);
imprimeNumerao(a, tamA); /* imprime 7 */

tamC = copiaNumerao(c, a, tamA);
tamC = divide(c, tamC, b, tamB);
imprimeNumerao(a, tamA); /* imprime 7 */
imprimeNumerao(b, tamB); /* imprime 3 */
imprimeNumerao(c, tamC); /* imprime 2 */
```

Lembre-se que sua estrutura de dados trabalha somente com inteiros. Sendo assim, divisões devem ser arredondadas.

Exemplos:

- $7 \div 2 = 3$
- $-5 \div 3 = -1$
- $7 \div -2 = -3$

Atenção: Vamos convencionar que as divisões devem sempre ser arredondadas em direção ao zero (truncadas)[1]. Lembre-se, tanto a sua operação de divisão quanto a operação de resto devem ser consistentes e obedecer à seguinte regra: $a = q \times b + r$, onde a é o dividendo, b o divisor, q é o quociente e r é o resto. Logo:

- $7 \ \% \ 2 = 1 = r$ já que $q = 7 \div 2 = 3$ e $7 = 3 \times 2 + 1$
- $-5 \ \% \ 3 = -2 = r$ já que $q = -5 \div 3 = -1$ e $-5 = -1 \times 3 + (-2)$
- $7 \ \% \ -2 = 1 = r$ já que $q = 7 \div -2 = -3$ e $7 = -3 \times -2 + 1$

Logo no início da sua implementação você vai reparar que existem dependências entre as operações. Por exemplo, para implementar a multiplicação você vai precisar da soma. Para implementar a divisão você vai precisar de subtração.

Sugestão para o desenvolvimento:

[1] Há uma grande discussão sobre como o resto da divisão, e consequentemente a direção do arredondadamento, deve ser calculado. Há quem defenda que o arredondamento deva ser feito em direção ao $-\infty$ enquanto outros argumentam que deva ser em direção ao 0. Veja https://en.wikipedia.org/wiki/Modulo_operation para mais informações.

- Comece trabalhando apenas com números positivos.
- Implemente a multiplicação. Primeiramente com números de um só dígito. Depois generalize para números de múltiplos dígitos. Teste à exaustão.
- Faça testes exaustivos na soma e multiplicação com números negativos e positivos.
- Só depois que tudo acima estiver funcionando perfeitamente parta para a divisão e resto. Estas são as operações mais complicadas de todas e dependem de tudo o que você fez antes. Então, para evitar dores de cabeça: teste com afinco as outras três operações!

3 Entradas e saídas

O seu programa deve ler os dados da entrada padrão e escrever os resultados na saída padrão. Para isto você pode usar as funções `scanf` e `printf`.

- A entrada será dada por uma expressão numérica dada em formato pós-fixo. Exemplos: "5 3 +" , "7 3 -" , "9 -4 /" , "6 7 *" .
- Cada um dos elementos da expressão estará em uma linha separada.
- Cada linha de entrada trará um número (tamanho arbitrário), uma operação ($+$, $-$, $*$, $/$, $\%$) ou a palavra "FIM". Por conveniência vamos limitar os números a 1000 dígitos (ou seja, 1001 caracteres no máximo quando incluirmos números negativos).
- Note que operações feitas com dois números de 1000 dígitos podem resultar em números com mais de 1000 dígitos. O limite de 1000 dígitos é apenas válido para os números dados como entrada. Não há limite quanto ao tamanho dos números obtidos como resultado de operações.
- Você pode assumir que a entrada é sempre uma expressão válida.
- Seu programa deve parar a execução quando receber a entrada "FIM".
- Toda vez que um operando for lido, a operação referente deve ser aplicada aos dois elementos especificados nas duas linhas anteriores do arquivo de entrada. O resultado da operação deve ser então impresso na tela.

Nos exemplos a seguir, entrada e saída aparecem intercaladas. O texto em vermelho indica a saída esperada.

3.1 Exemplo 1

```
3
5
+
8
8
2
/
4
4
3
%
1
1
5
*
5
5
-1
-
6
FIM
```

3.2 Exemplo 2

```
2
3
*
6
6
4
*
24
24
5
*
120
120
6
*
720
720
7
```

```
*
5040
5040
8
*
40320
40320
9
*
362880
362880
10
*
3628800
FIM
```

3.3 Exemplo 3

```
98745265432294533098524308943230863 22
3251422875648373547423311878720900987 1
*
3210626148885330266935799054403256517114126953018942893077398143184830844 62
3210626148885330266935799054403256517114126953018942893077398143184830844 62
9
/
3567362387650366963261998949336951685682363281132158770085997936872034271 8
FIM
```

3.4 Implementação de referência

Uma implementação de referência está disponível (apenas o binário :p). Você pode baixá-la juntamente com os testes abertos nos links abaixo:

- Implementação de referência
  - Versão Linux 64.
  - Versão Windows (32 bits).
- Testes abertos.

4 Entrega e avaliação

Este projeto poderá ser feito em grupos de até três pessoas. A entrega deverá ser feita pelo Moodle **apenas** por um dos integrantes do grupo. O arquivo enviado deve conter um cabeçalho contendo o nome completo e RA de cada

um. Verifique a página da disciplina para informações sobre as datas de entrega e acesso ao Moodle.

Todos os trabalhos entregues passarão por uma correção automática no Moodle. A nota será proporcional ao número de testes que passarem com sucesso. Contudo, o monitor e os professores poderão, ao seu próprio critério, alterar as notas para cima ou para baixo. Passar na metade do testes não significa que você tirou nota 5 automaticamente, indica apenas a provável nota que será atribuída ao seu trabalho.

Bônus: Faremos uma competição considerando todos os alunos das três turmas. Os programas que forem mais rápidos ganharão nota extra na média geral do projeto[2] conforme as regras abaixo:

- Um ponto a mais para o trabalho mais rápido da turma.
- Dois pontos a mais para o trabalho mais rápido entre as 3 turmas.

ATENÇÃO: Plágios serão severamente punidos com a reprovação na disciplina.

4.1 Política de atrasos

Trabalhos entregues com atraso serão aceitos, porém sofrerão uma redução na sua nota máxima conforme tabela abaixo:

Dias em Atraso	**Nota Máxima**
0	10
1	7
2	6
3	5
>3	0

Bom trabalho!

[2]Pergunte ao seu professor como você pode ganhar desempenho na implementação da sua solução! Por exemplo, como fazer um melhor uso da memória utilizando alocações dinâmicas ou ainda utilizar melhores algoritmos para uma determinada tarefa. Em particupar, existe um algoritmo para multiplicar números com muitos dígitos mais rápido do que o tradicional que utilizamos com papel e lápis chamado Karatsuba (https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/karatsuba.html).