Departamento de Ciência da Computação – IME-USP

MAC 115 – Introdução à Computação para Ciências Exatas e Tecnologia

Escola Politécnica – Primeiro Semestre de 1997

Terceiro Exercício-Programa — Data de entrega: até 20 de maio de 1997.

Turmas: Computação; Elétrica A; Elétrica B; Elétrica C; Elétrica D; Mecatrônica.

CALCULADORA

O objetivo deste exercício-programa é construir uma biblioteca de funções para serem utilizadas na confecção de uma calculadora que trabalhe com números inteiros grandes. Para isso, os números não podem ser armazenados em variáveis do tipo `int` ou mesmo `long int`, já que estas armazenam valores de magnitude limitada. Uma técnica é utilizar para cada número um vetor que contenha em cada componente um dígito do número.

O comportamento do programa, do ponto de vista de quem usa, deverá ser semelhante a uma calculadora de bolso do tipo mais simples. Nesta primeira parte, você deverá fazer algumas funções que serão utilizadas para fazer essa calculadora. Os detalhes finais para a confecção da calculadora serão dados no próximo exercício-programa.

Alguns detalhes sobre esta primeira parte

(1) O número máximo de dígitos deve ser fixado através de uma constante (por exemplo, 60). Para isso, coloque no início do seu programa

```
# define MAX_DIGITOS 60
```

(2) O sinal do número deve ser guardado na posição zero do vetor. Considere a seguinte convenção: se o número é positivo, coloque 0 (zero) nessa posição; caso contrário coloque 1 (um). Por exemplo, o número `-321938492382734` será armazenado em um vetor como

1	0	0	···	0	3	2	1	9	3	8	4	9	2	3	8	2	7	3	4

Note que o número de casas de um vetor que armazena um número deve ser `MAX_DIGITOS+1`, devido à casa reservada para o sinal.

(3) A entrada e a saída serão feitas através de variáveis do tipo string. Para a leitura de um número, o fim desse número será indicado através do `<ENTER>`.

As funções que você deve implementar

Neste exercício-programa, você deve construir **obrigatoriamente** as seguintes funções:

1. `void zera (int numero[]);`

   Preenche o vetor `numero` com zeros.

2. `int soma (int num1[], int num2[], int res[]);`

   Soma o vetor `num1` ao vetor `num2` (considere que ambos os números sejam positivos) e armazena o resultado no vetor `res`. No caso de ser detectado overflow (ocorreu um vai um quando da soma dos dígitos mais significativos) a função deve retornar valor 1 (um). Caso contrário, a função deve retornar valor zero.

3. void `subtrai (int num1[], int num2, int res[]);`

   Subtrai o vetor `num1` do vetor `num2`, `num1 - num2`, (considere que ambos os números sejam positivos) armazena o resultado no vetor `res`.

4. `int multiplica (int num1[], int num2[], int res[]);`

   Multiplica o vetor `num1` pelo vetor `num2` e armazena o resultado no vetor `res`. Para fazer a multiplicação, utilize o algoritmo usual de multiplicar dois números (é claro que você terá que usar a função `soma`). No caso de ser detectado overflow (ocorreu um vai um quando da soma dos dígitos mais significativos) a função deve retornar valor 1 (um). Caso contrário, a função deve retornar valor zero.

5. `int quociente (int num1[], int num2[], int res[]);`

   Divide o vetor `num1` pelo vetor `num2` e armazena o quociente no vetor `res`. Para fazer a divisão, faça subtrações sucessivas utilizando a função `subtrai`. No caso do vetor `num2` ser zero, a função deve retornar valor 2 (dois). Caso contrário, a função deve retornar valor zero.

6. `int resto (int num1[], int num2[], int res[]);`

   Calcula o resto da divisão do vetor `num1` pelo vetor `num2` e armazena o resultado no vetor `res`. Para encontrar o resto, faça subtrações sucessivas utilizando a função `subtrai`. No caso do vetor `num2` ser 2 (dois), a função deve retornar valor zero. Caso contrário, a função deve retornar valor zero.

7. `int maior_igual (int num1[], int num2[]);`

   Essa função devolve um se `num1` $\geq$ `num2` e zero caso contrário.

8. `void reverte (int num[]);`

   reverte o sinal do vetor `num`.

9. `void copia_num (int num1[], int num2[]);`

   Copia o vetor `num2` em `num1`.

Funções para entrada e saída

Para a entrada e saída de dados, faça obrigatoriamente as seguintes funções:

1. `void pega_entrada (char str[]);`

   Lê um string do teclado e guarda em `str`.

2. `void coloca_tela (char str[]);`

   Imprime o string `str` na tela.

3. `int desempacota (char str[], int num[]);`

   Transforma os caracteres do string `str` em números transferindo-os para o vetor `num` tomando o cuidado de verificar se o primeiro caractere é um sinal (`´+´` ou `´-´`) ou um dígito (`´0´` a `´9´`) e todos os demais caracteres dígitos. Note que em `str` deve-se ter pelo menos um dígito. Se houver alguma inconsistência, a função deverá retornar valor 3 (três), caso contrário, deverá retornar valor zero.

4. `void empacota (int num[], char str[]);`

   Transforma os números do vetor `num` em caracteres para o string `str` tomando o cuidado de colocar o sinal corretamente.

Para este exercício-programa, faça uma tela inicial da seguinte forma:

```
CALCULADORA (FASE INICIAL)

     01.  Somar
     02.  Subtrair
     03.  Multiplicar
     04.  Quociente
     05.  Resto
     06.  Maior_Igual
     07.  Reverte
     08.  Copia
     09.  Zera
     10.  Sair

     opcao:  _
```

Depois de feita a escolha, seu programa ou deve ler dois números (opções de 1 a 6) ou ler um número (opções de 7 a 9) ou terminar (opção 10). Além disso, se a opção está entre 1 e 9, seu programa deve imprimir o resultado da operação (se ocorreu um erro, então o resultado é a mensagem de erro correspondente).

Faça também uma saída para arquivo, de forma que você possa seguir todas opções que foram realizadas pelo seu programa quando executado. Isto funcionará como um pequeno relatório gerado pelo seu programa e facilitará a correção do mesmo (você deverá imprimi-lo para entregá-lo juntamente com um disquete e uma listagem de seu programa).

Tratamento de erro

`void trata_erro (int tipo_do_erro);`

Esta função trata uma situação de erro, imprimindo a mensagem correspondente ao erro. Considere os seguintes códigos de erro:

`tipo_do_erro`	erro
01	overflow
02	divisão por zero
03	entrada não é número

Faça pelo menos este teste!

Faça quantos testes você achar necessário, mas execute pelo menos estes que se seguem e entregue-os juntamente com a listagem do seu programa.

a. 99999999999999999999999999999999999999999999999999999999999 + 3.

b. 123456789123456789 + 123456789.

c. 123456789123456789 − 987654321.

d. 98765432 − 123456789.

e. 1234567 × 987654321.

f. −23456789 × 987654.

g. 987654321987654321987654321987654321987654321987654321 × 987654321.

h. 987654321 / 123456789.

i. 87654321 / −23456789.

j. 12345678 / 0.

k. 987654321 mod 123456789.

l. 98765432 mod 0.

m. −987654321 é maior ou igual a 123456789 ?

n. 987654321 é maior ou igual a 24678912 ?

o. 123456789 é maior ou igual a 987654321?

p. 123456 é maior ou igual a 123456?

q. reverta o sinal de 987654.

r. copie o número 987654321.

s. leia o número 987612345 e "zere-o".

t. tente ler 987po1209sfa e seu programa deverá dar uma mensagem de erro.