Departamento de Ciência da Computação – IME-USP

MAC211 – Laboratório de Programação I
Março de 2013

Primeiro Exercício-Programa (EP1)
Data de entrega: 07/04/2013

1 Introdução: Números de Armstrong

Um número é chamado de *número de Armstrong* quando a soma de cada um do seus dígitos elevado à quantidade de dígitos do número equivale ao próprio número. Por exemplo, 153 e 93084 são números de Armstrong já que:

$$\begin{aligned} 153 &= 1^3 + 5^3 + 3^3 \\ 93084 &= 9^5 + 3^5 + 0^5 + 8^5 + 4^5 \end{aligned}$$

Os números de Armstrong existentes no intervalo [1..100000000] são:

0	1	2	3	4	5	6	7	8	9
153	370	371	407	1634	8208	9474	54748	92727	93084
548834	1741725	4210818	9800817	9926315	24678050	24678051	88593477		

2 Descrição do Exercício-Programa

2.1 Parte em linguagem de montagem

Implemente em linguagem de montagem uma função que tem o seguinte protótipo

```
int seleciona_armstrongs(char* nome_arq_entrada, char* nome_arq_saida);
```

A função deve receber como parâmetro duas cadeias de caracteres contendo o nome de um arquivo texto de entrada e o nome de um arquivo texto de saída para a função. O arquivo de entrada contém números positivos de 32 bits (ou seja, valores entre 0 e 4294967295), separados entre si por um caracter de espaço. Sua função deve abrir e ler o arquivo de entrada, verificando se cada número lido é um número de Armstrong. Se o número for de Armstrong, você deve gravá-lo no arquivo de saída. Sua função deve cuidar da criação do arquivo de saída caso ele não exista. Além disso, sua função deve devolver como valor de retorno a quantidade de números de Armstrong encontrados no arquivo de entrada.

Importante: as únicas chamadas a funções das bibliotecas de C que você pode fazer na parte em linguagem de montagem são para funções de manipulação de arquivos (abertura, leitura, escrita, etc.), ou para funções de leitura da entrada padrão ou escrita na saída padrão.

2.2 Parte em linguagem C

Implemente em linguagem C um programa que lerá os dados de entrada da entrada padrão (teclado ou arquivo redirecionado), chamará a função em linguagem de montagem e imprimirá o resultado na saída padrão. Por exemplo, se a entrada padrão receber os seguintes parâmetros

```
arquivoLeitura.txt arquivoEscrita.txt
```

e se `arquivoLeitura.txt` contiver os seguintes inteiros

```
918211 8934701 548834 2356 12 3216 1634 78523009 232
```

então o programa precisa imprimir na saída padrão:

```
2 numero(s) de Armstrong encontrado(s).
```

e o arquivo `arquivoEscrita.txt` deverá conter os números:

```
548834 1634
```

2.3 Avaliação do desempenho de sua implementação

Escreva uma segunda versão de seu programa, na qual a função `seleciona_armstrongs` é implementada em linguagem C. Depois, compare o desempenho das duas implementações da função quando executadas sobre um mesmo arquivo de entrada.

Para medir o tempo que a sua função leva para ser executada, você pode usar funções da biblioteca de C `time.h`. O exemplo a seguir mostra como fazer isso.

```
#include <stdio.h>
#include <time.h>

int main()
{
    clock_t inicio, fim;
    double tempo_execucao;

    inicio = clock();    /* marca o horario de inicio da execucao */

    minha_funcao();      /* chama a funcao cujo tempo de execucao sera' medido */

    fim = clock();       /* marca horario de fim da execucao */
    tempo_execucao = (double)(fim - inicio) / CLOCKS_PER_SEC;
    printf("A execucao da funcao levou %f segundos \n", tempo_execucao);

    return 0;
}
```

Avalie o tempo de execução da função para arquivos de entrada contendo as seguintes quantidades de números: 100 mil, 1 milhão, 10 milhões, 50 milhões e 100 milhões.

Faça um programa para gerar os arquivos texto de entrada (com os tamanhos mencionados acima) usados na avaliação de desempenho de suas funções. Gere dois tipos de arquivos: arquivos contendo números sequenciais (por exemplo, de 1 a mil) e arquivos contendo números aleatórios. Na página `http://www.ime.usp.br/~pf/algoritmos/aulas/random.html`, o prof. Paulo Feofiloff explica didadicamente como gerar números aleatórios em C. Não se esqueça de que o arquivo de entrada deve somente conter números positivos de até 32 bits.

Escreva um relatório simples, com uma tabela apresentando os tempos de execução que você observou em seus experimentos. Em seu relatório, aponte motivos que justifiquem a possível diferença de desempenho entre as duas versões da implementação da função. **Dica:** usando o GCC, gere o código em linguagem de montagem para a versão em C da função. Depois, compare-a com a sua implementação em linguagem de montagem.

Como o tempo de execução da função pode variar a cada chamada, execute 10 vezes a função para cada arquivo de entrada considerado na análise. Coloque na tabela do seu relatório a média dos tempos obtidos para cada arquivo de entrada. Inclua também no relatório as características da máquina (tipo do processador, velocidade, etc.) na qual foram executados os experimentos.

Observação 1: ao executar seu programa usando como entrada os arquivos com números aleatórios, não se espante se a quantidade de números de Armstrong encontrados no arquivo for nula. A probabilidade de se encontrar um número de Armstrong entre os números gerados aleatoriamente é bem pequena. Para verificar se sua função está realmente correta, use arquivos de entrada contendo números sequenciais. A Seção 1 listou todos os números de Armstrong existentes no intervalo [0..100000000]

Observação 2: A geração de um arquivo de entrada com 50 ou 100 milhões de números levará alguns minutos para ser executada. Entretanto, a "busca" por números de Armstrong nesses arquivos leva bem menos tempo.

3 Considerações Finais

- Você deve entregar via Paca um único arquivo compactado, contendo todos os arquivos dos códigos-fonte que você criou para a resolução do EP (o que inclui o código para a geração dos arquivos de entrada usados na avaliação de desempenho).

- A parte em linguagem de montagem deve ser desenvolvida usando a sintaxe da Intel, para uma arquitura de 32 bits (i386) e sistema operacional Linux. Para a geração do código objeto da parte em linguagem de montagem, você pode usar o NASM ou o GCC. Para a ligação (geração do código executável), use o GCC.

- Um dos critérios de avaliação do EP será a organização do código. Portanto, divida o seu código em sub-rotinas e funções sempre que for conveniente e documente-o apropriadamente. Inclua na documentação instruções sobre: (i) como o seu código deve ser montado e ligado (para a geração do executável); (ii) como os parâmetros de entrada do(s) programa(s) devem ser passados na linha de comando.

- O EP deve ser realizado em duplas. Entretanto, em casos excepcionais e bem justificados, trabalhos individuais serão aceitos.
  Importante: cada componente da dupla deve participar de todas as etapas da solução do exercício.
  Sugestão: cada dupla pode trabalhar lado a lado, no mesmo computador (exercitando o conceito de *programação pareada*).