CCM0118 — Computação I

Curso de Ciências Moleculares — Turma 22 — Segundo Semestre de 2012

Primeiro Exercício-Programa — Data de entrega: até **17 de setembro de 2012.**

Conversão de Bases

Escreva um programa em C que converta números inteiros não negativos de uma base para outra. O programa deve funcionar com bases de 2 a 36. Para as bases maiores que 10, o programa usará como dígitos adicionais os caracteres '`a`', '`b`', '`c`', ... Esses caracteres representarão os números 10, 11, 12, ... No caso da base 36, o dígito de maior valor será '`z`', que representará o número 35.

O exemplo abaixo mostra como seu programa deverá funcionar. Nesse exemplo, os números que aparecem sublinhados foram digitados pelo usuário. Todo o resto é a saída do programa.

```
Base (entre 2 e 36) do número a ser convertido, ou 0 para sair: 16
                            Número na base 16 a ser convertido: ffff
  Base (entre 2 e 36) para a qual o número deve ser convertido: 10
               Resultado da conversão do número para a base 10: 65535

Base (entre 2 e 36) do número a ser convertido, ou 0 para sair: 10
                            Número na base 10 a ser convertido: 1034831966394
  Base (entre 2 e 36) para a qual o número deve ser convertido: 16
               Resultado da conversão do número para a base 16: f0f0cab0ba

Base (entre 2 e 36) do número a ser convertido, ou 0 para sair: 32
                            Número na base 32 a ser convertido: 19626lklnfo1
  Base (entre 2 e 36) para a qual o número deve ser convertido: 36
               Resultado da conversão do número para a base 36: computacao1

Base (entre 2 e 36) do número a ser convertido, ou 0 para sair: 8
                             Número na base 8 a ser convertido: 310232045474
  Base (entre 2 e 36) para a qual o número deve ser convertido: 36
               Resultado da conversão do número para a base 36: ccm0118

Base (entre 2 e 36) do número a ser convertido, ou 0 para sair: 0
```

Requisitos

- O programa deve encerrar sua execução quando for digitado um zero no lugar da base do número a ser convertido.
- A saída do seu programa deve ser formatada como o exemplo acima. Haverá só uma diferença visual: os valores digitados pelo usuário não aparecerão sublinhados. (O exemplo mostra esses valores sublinhados apenas para distinguir o que é entrada do programa e o que é saída.)
- Quando estiver lendo o número a ser convertido, o programa não fará distinção entre maiúsculas e minúsculas. Em outras palavras, os dígitos adicionais podem também ser representados pelos caracteres '`A`', '`B`', '`C`', ...

- O número a ser convertido deve ser lido caractere a caractere. A leitura de cada caractere do número deve ser feita por meio da função `getchar` de `<stdio.h>`. (É possível ler os caracteres do número usando `scanf` com `"%c"`, mas essa não é a melhor opção.) De modo análogo, a impressão do resultado da conversão de bases deve ser feita caractere a caractere. A impressão de cada caractere do resultado deve ser feita por meio da função `putchar` de `<stdio.h>`. Já as bases podem ser lidas com `scanf`.

- O número a ser convertido deve ser lido caractere a caractere e armazenado numa variável do tipo `unsigned long long`. O programa primeiro converterá toda a sequência de caracteres do número para um `unsigned long long`. Depois de ter esse `unsigned long long`, o programa o imprimirá na nova base.

- O programa deve ser capaz de trabalhar com números grandes. O maior valor que cabe numa variável `unsigned long long` é 18446744073709551615. Seu programa deve funcionar para números próximos desse valor.

- (Este item é para quem já conhece a linguagem C.) O programa não deve usar vetores. Em particular, os caracteres do número a ser convertido não devem ser armazenados num vetor.

- O programa deve ser compilável com o seguinte comando:

  ```
  gcc -std=c99 -pedantic -Wall -O2 -U_FORTIFY_SOURCE
  ```

  A compilação deve ocorrer sem erros nem *warnings*.

  Note que neste EP você usará o gcc com a opção `-std=c99` em vez da opção `-ansi` usada no EP0. Há um motivo para isso: Os tipos `long long` e `unsigned long long` foram introduzidos na padronização de 1999 da linguagem C. Eles não existiam na padronização anterior (o padrão ANSI C, de 1990).

Recomendações

- Um exemplo de solução deste EP (um programa executável no Linux) está disponível na página da disciplina. Nos casos em que este enunciado for omisso, faça seu programa imitar a solução exemplo.

- O uso de funções facilitará muito seu trabalho e fará seu programa ficar bem mais organizado. Recomendo fortemente que, além da função `main`, seu programa tenha pelo menos as duas funções descritas abaixo.

  `unsigned long long read_number(unsigned base);`
  : Lê uma sequência de caracteres que representa um número na base especificada pelo parâmetro `base` e devolve um `unsigned long long` com o número lido.

  `void write_number(unsigned long long n, unsigned base);`
  : Imprime uma sequência de caracteres com a representação do número dado pelo parâmetro `n` na base especificada pelo parâmetro `base`.

Instruções de entrega e informações adicionais importantes

Serão colocadas aqui em breve...

Bom trabalho!