Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Verão de 2018/2019

Série de Exercícios 1

Realize os exercícios seguintes usando a linguagem C. Não se esqueça de testar devidamente o código desenvolvido, bem como de o apresentar de forma cuidada, apropriadamente indentado e comentado. Assegure-se de que o compilador não emite qualquer aviso sobre o seu código, mesmo com a opção `-Wall` activa. Contacte o docente se tiver dúvidas. Não é necessário relatório. Encoraja-se a discussão de problemas e soluções com outros colegas, mas a partilha directa de soluções leva, no mínimo, à anulação das entregas de todos os envolvidos.

1. Implemente a função **`string_detab`**, que substitui os caracteres TAB (**`'\t'`**) por caracteres espaço na *string* recebida no parâmetro **`string`**. O parâmetro **`tab_size`** define a dimensão da tabulação. Considere que a variável utilizada como argumento em **`string`** tem dimensão suficiente para o aumento de caracteres na *string*.

   ```
   void string_detab(char *string, int tab_size);
   ```

2. Considere vectores de *bits* representados sobre *arrays* de inteiros e que **`INT_BIT`** é o número de *bits* de um inteiro. No valor de índice **`n`** de um *array* de inteiros, o *bit* de menor peso corresponde ao índice **`n * INT_BIT`** do vector de *bits* e o *bit* de maior peso corresponde ao índice **`(n + 1 ) * INT_BIT - 1`** do vector de *bits*. A função **`vgetbits`** retorna o valor dos *bits* entre as posições **`idx`** e **`idx + len - 1`** do vector de *bits* representado por **`data`**. A função **`vsetbits`** escreve os **`len`** *bits* de menor peso de **`val`** nas posições entre **`idx`** e **`idx + len - 1`** do vector de *bits* representado por **`data`**. Em ambas as funções, **`len`** nunca é maior do que **`INT_BIT`**. Ex: para **`data = { 0xABCD1234, 0xFFFFFFEC }`**, a chamada a **`vgetbits(data, 29, 8)`** retorna **`0x00000065`**. Defina **`INT_BITS`** e escreva as funções **`vgetbits`** e **`vsetbits`**.

   ```
   unsigned int vgetbits(unsigned int data[], unsigned int idx, unsigned int len);
   void vsetbits(unsigned int data[], unsigned int idx, unsigned int len, unsigned int val);
   ```

3. Escreva a função **`string_to_float`**, para converter um valor real, recebido no parâmetro **`string`**, para o formato IEEE 754. O resultado da conversão deve ser retornado como valor da função. Como formatação do argumento considere apenas a notação portuguesa constituída por parte inteira, seguida de vírgula e da parte decimal. Na implementação interna só podem ser utilizadas operações aritméticas e lógicas sobre inteiros. Qualquer operação de vírgula flutuante invalida a resolução do exercício.

   ```
   float string_to_float(char *string);
   ```

4. Considere a utilização do tipo **`struct citizen_node`** para registo de informação sobre eleitores numa freguesia. Escreva a função **`group_by_polling_place`**, que recebendo o conjunto de todos os eleitores, no *array* **`all`**, com dimensão **`all_num`**, agrega os eleitores por local de voto (**`polling_place`**). A agregação é concretizada criando uma lista simplesmente ligada relativa a cada local, utilizando o campo **`next`**. As listas são referenciadas no *array* **`places`**, em que cada índice do *array* corresponde directamente ao número do local de voto e cada posição contém um ponteiro para o primeiro nó da respectiva lista. À entrada na função os valores dos campos **`next`** são indefinidos.

   ```
   typedef struct citizen_node {
       const char *name, *civil_id_no;
       unsigned polling_place;
       struct citizen_node *next;
   } Citizen_node;
   void group_by_polling_place(Citizen_node *places[], Citizen_node all[], size_t all_num);
   ```

5. Realize o programa utilitário **`mygrep`**, que escreve no *standard output* as linhas lidas do *standard input* que contenham alguma das palavras passadas como argumento ao programa. Se, no conjunto de argumentos, for incluída a opção **`-r`**, a escrita das linhas deve ser feita por ordem inversa à que são lidas do *standard input* (a linha escrita em primeiro lugar é a última linha a ser lida que contém uma das palavras indicadas).

Data recomendada de entrega: 31 de Março de 2019

ISEL, 7 de Março de 2019