Lista de exercícios
ACSL, Frama-C e Wp

Richard Bonichon Vítor Almeida

20141002

Introdução

Esta lista propõe exercícios para compreensão e aprendizagem do uso das ferramentas **Frama-c** e **Wp** vistos em sala de aula.

As especificações exigidas em todas as questões devem ser em formato ACSL. Para a verificação destas especificações podem ser usados tanto a versão de console quanto a versão com interface gráfica do frama-c. Para a versão do console, basta executar o comando

```
% frama-c -wp -wp-rte -pp-anot arquivo.c
```

Substituindo *arquivo.c* pelo arquivo a ser verificado. A opção `-pp_anot` permite a leitura de dados em diretivas `#includes` como, por exemplo, obter o valor máximo e mínimo de números inteiros. A opção `-wp-rte` permite uso do *plugin* **rte**, que gera anotações para verificar acesso válido de memória, *overflow,underflow*, etc.

Na versão com interface gráfica, o arquivo pode ser diretamente aberto e verificado substituindo-se o comando `frama-c` por `frama-c-gui`.

Algumas referências que podem e devem ser consultadas são:

- *ACSL by Example*

  `http://www.fokus.fraunhofer.de/de/sqc/_download_sqc/ACSL-by-Example.pdf`

- Manual do ACSL

  `http://frama-c.com/download/acsl_1.8.pdf`

- Manual do plugin Wp

  `http://frama-c.com/download/frama-c-wp-manual.pdf`

Instruções

Cada exercício deve ser armazenados em pastas denominadas lab_N, onde N deve ser substituído pelo número do exercício e, para o código-fonte, criar arquivos lab_N.c, substituindo L pela letra da subquestão seja este o caso.

1 Max

1. Escreva a especificação e implemente uma função que leia dois números reais e retorne o maior deles. Esta função deve ser denominada `max`.

2. Que modificações no código e na especificação devem ser feitas se os parâmetros forem ponteiros? Refaça a função e sua especificação com a seguinte declaração:

   ```
   int max(int* p, int* q)
   ```

2 Abs

Especifique e implemente a função:

```
int abs(int x)
```

no qual, dado um valor inteiro x, a função retorna o valor absoluto do mesmo.

3 Número primo

Escreva a especificação e implementação de uma função que verifique se um dado número inteiro é primo.

A declaração da função deve ser:

```
int primo(int n)
```

4 Cubo

1. Escreva uma função que lê um número de ponto flutuante e produza como saída o cubo do mesmo. A declaração da função deve ser:

   ```
   int cube(int x)
   ```

2. Reescreva a mesma função, porém o parâmetro é enviado por referência e o valor do resultado é armazenado na mesma variável, retornando *void*. A declaração da função deve ser:

   ```
   void cube(int* x)
   ```

5 MDC

Escreva a especificação e implementação de uma função que calcula e retorna o MDC (Máximo Divisor Comum) de dois números inteiros.

A declaração da função deve ser:

```
int mdc(int a, int b)
```

6 Find array

O código abaixo servirá para responder as questões 6.1 e 6.2:

```c
/*
[find_array(arr,length,query)] retorna o índice [idx]
do array ordenado [arr] de tamanho [len]
tal que [arr[idx] == query].

Se tal índice não existe, retorna -1
*/
int find_array(int* arr, int len, int query);
 int find_array(int* arr,int length, int query) {
   int low = 0;
   int high = length - 1;
   while  (low <= high) {
     int  mean = low + (high -low) / 2;
     if (arr[mean] == query)
     return  mean;
     if (arr[mean] < query) low = mean + 1;
     else high = mean - 1;
   }
   return -1;
 }
```

6.1 Pré e pós-condições

1. Escreva a especificação formal ACSL da função `find_array`. Pontos a considerar:

   - O tamanho de `arr` deve ser positivo ou 0,
   - `arr` deve conter pelo menos `len` memórias válidas que podem ser seguramente lidos,
   - `arr` deve ser ordenado,
   - Se `arr` contém `query` em algum índice entre `0` e `len - 1`, o valor retornado pela função deve estar neste intervalo, assim como `arr[\result]` deve ser igual a `query`,
   - Se todos os elementos do `arr` forem diferentes de `query`, o valor retornado deve ser -1.

2. Reescreva a especificação anterior usando dois comportamentos distintos no qual um corresponde ao caso em que o elemento `query` é encontrado enquanto que o outro ocorre em caso contrário.

3. Reescreva a especificação definida em 2 usando dois predicados lógicos:

   - `sorted`: verifica se um dado `arr` está ordenado,
   - `mem`: verifica se um dado elemento está contido em `arr`

6.2 Laço

1. Escreva a especificação do laço no código `find_array`. Pontos a considerar:

   - assegurar que não será acessada nenhuma posição fora dos limites de `arr`,
   - assegurar que para todos os elementos no intervalo [0..`low`) de `arr` são inferiores a `query` enquanto que todos os elementos no intervalo (`high`..`len`) são superiores ao mesmo;
   - assegurar que o laço sempre termina,
   - definir quais variáveis externas ao laço serão modificadas pelo mesmo

2. Verifique com o plugin `Wp` de `Frama-C` que as suas especificações desta e da questão 6.1 estão de acordo com as asserções definidas na função `main` abaixo:

```
void main () {
  int array[] = { 0, 4, 5, 5, 7, 9 };
  int idx = find_array(array,6,7);
  /*@ assert idx == 4; */
  idx = find_array(array,6,5);
  /*@ assert idx == 2 || idx == 3; */
  idx = find_array(array,5,9);
  /*@ assert idx == -1; */
  array[0] = 6;
  idx = find_array(array,4,6);
}
```

7 Substituição no *array*

Especifique e implemente uma função que, dado um parâmetro x, retornar um *array* com 10 elementos no qual, para cada posição i, o valor desta posição deve ser (x*i)/2. A declaração da função deve ser:

```
int* subs_array(int x)
```

8 Maiúscula

Escreva a especificação de uma função que transforma todos as letras minúsculas em maiúsculas de um vetor de caracteres. Letras já maiúsculas e outros símbolos são ignorados. A declaração da função é:

```
void maiuscula(char * v, int len)
```

1. Utilize estratégias diferentes para especificação e implementação - se na especificação são feitas comparações entre caracteres, faça comparação com inteiros do código ASCII na implementação ou vice-versa
2. Caso seja feito um laço, insira a especificação do mesmo
3. Se o provador pode não provar todas as obrigações de prova, tente minimizar para no máximo uma ou duas obrigações de prova não concluídas - sem deixar a especificação incompleta

9 Detecção de subcadeia de caracteres

Escreva a especificação e implementação de uma função que, dados dois vetores de caracteres `string` e `sub`, verifica se `sub` corresponde aos últimos caracteres de `string`.

Por exemplo, "eco" corresponde ao final de "caneco". A declaração da função é:

```
int subsfinal(char* string, int l, char* sub, int t)
```

10 Ordenação por inserção

Dado o algoritmo a seguir, faça a especificação da função e dos dois *laços* internos:

```
 void insertionSort(int numeros[], int tam) {
   int i, j, eleito;

   for (i = 1; i < tam; i++){
      eleito = numeros[i];
      j = i - 1;
      while ((j>=0) && (eleito < numeros[j])) {
         numeros[j+1] = numeros[j];
         j--;
      }
      numeros[j+1] = eleito;
   }
}
```