Lista de exercícios de testes de *software*
Testes Caixa Preta

Richard Bonichon Vítor Almeida

201411

Introdução

Esta lista propõe exercícios para compreensão e aprendizado de testes caixa preta e caixa branca vistos em sala de aula.

Para isto, será utilizada a ferramenta *Cunit* (`http://cunit.sourceforge.net/`)

Instruções

Cada exercício deve ser armazenados em pastas denominadas lab1_N, onde N deve ser substituído pelo número do exercício e, para o código-fonte, criar arquivos lab2_N.c ou lab2_N_L.c, substituindo L pela letra da subquestão seja este o caso.

1 Ordenação

Uma equipe implementou e disponibilizou como biblioteca estática uma função que ordena em ordem não-decrescente um vetor de inteiros. Porém, esta função deve respeitar certas características:

1. O vetor deve possuir entre 4 e 10 entradas (inclusive)
2. Cada número deve possuir 5 dígitos

3. a função retorna 0 caso a entrada não respeite as condições de entrada e 1 caso contrário

4. Só são aceitos números inteiros

Com isto, seu objetivo será de criar casos de testes de acordo com critérios de classes de equivalência, valor limite e testes funcionais sistemáticos. O casos de testes deverão ser executados com o *Cunit*.

A biblioteca estática está disponibilizada em dois arquivos:

- `libordenar.a`
- `ordenar.h`

Para compilá-los junto com seu código, execute o seguinte comando:

```
gcc -static -L. -lordenar -lcunit -o <binario> <arquivo.c>
```

A declaração da função de ordenação encontra-se no cabeçalho.

2 Substring

A mesma equipe também disponibilizou como biblioteca a função:

```
char* substring (char *str, int b)
```

Esta função retorna a cadeia de caracteres contendo os caracteres da cadeia de caracteres de entrada contidos entre a posição `b` (inclusive) e o último caractere. Caso `b` se encontre fora dos limites de `str`, é retornado uma cadeia de caracteres vazia (“´´).

Exemplos:

- substring(“hamburguer”, 4) == “urguer”
- substring(“programação modular”, 1) == “rogramação modular”
- substring(“vazio”, 5) == “”

Similarmente à questão anterior, você deverá criar e executar casos de testes no *Cunit* de acordo com critérios classes de equivalência, valor limite e testes funcionais sistemáticos.

Os arquivos desta biblioteca são:

- `libsubstring.a`
- `substring.h`

A declaração da função de ordenação encontra-se no cabeçalho.

3 Qual é o tipo do triângulo

Uma terceira função que deve ser testada lê três lados de um triângulo e imprime que tipo de triângulo é este. Sua declaração é:

```
int tipo_triangulo(double l1,double l2,double l3}
```

Alguns pontos frisados sobre esta questão:

1. A saída da função é um inteiro representando os seguintes valores:
   (a) 1 - Equilátero
   (b) 2 - Isósceles
   (c) 3 - Escaleno
   (d) 4 - Não é triângulo
   (e) 0 - Entrada inválida
2. Os parâmetros devem estar na faixa 0..99 (ou será entrada inválida)
3. Mesmo que triângulo equilátero seja isósceles a função deverá retornar 2

Similarmente à questão anterior, você deverá criar e executar casos de testes no *Cunit* de acordo com critérios classes de equivalência, valor limite e testes funcionais sistemáticos.

Os arquivos desta biblioteca são:

- `libtri.a`
- `tri.h`

A declaração da função de ordenação encontra-se no cabeçalho.