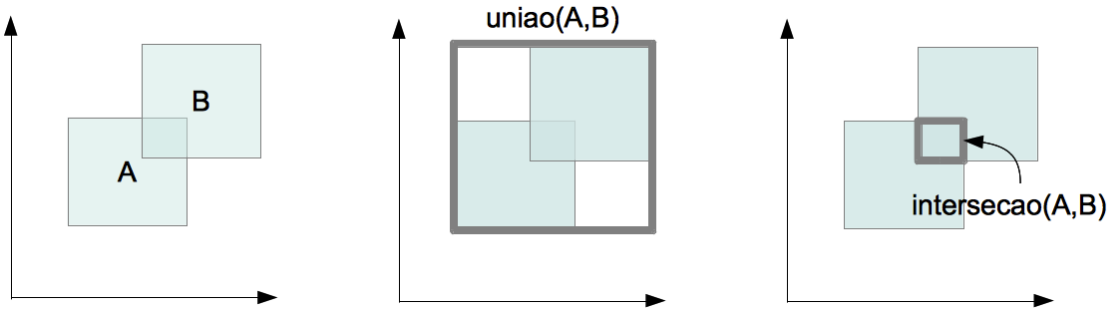
Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
AD1 - 2° semestre de 2013

---

1. (4 pontos) Um retângulo envolvente é um retângulo cujos lados são alinhados com os eixos coordenados usados para estimar a posição e tamanho de objetos bidimensionais em geral. Para tanto, é útil considerar as operações de união e interseção de retângulos envolventes (veja a figura abaixo). A *união* de dois retângulos envolventes é o menor retângulo envolvente que contém ambos os retângulos, enquanto que sua *interseção* corresponde ao maior retângulo cujos pontos pertencem a ambos os retângulos (pode ser um retângulo nulo).



Pede-se escrever quatro funções conforme descrito abaixo:

(a) A função retangulo($xmin,$xmax,$ymin,$ymax) retorna um array que representa um retângulo envolvente cujas dimensões x e y correspondem ao intervalo [$xmin,$xmax] e [$ymin,$ymax], respectivamente.

Gabarito

```
function retangulo($xmin,$xmax,$ymin,$ymax) {
    return array(array($xmin,$xmax),array($ymin,$ymax));
}
```

(b) A função formata($a) retorna uma string representativa do retângulo $a (array produzido pela função retangulo). Em particular, a string deve conter as dimensões do retângulo caso este possua área não nula, caso contrário, deve ser igual a "Retangulo nulo".

Gabarito

```
function format($r) {
    if ($r[0][0]>$r[0][1] || $r[1][0]>$r[1][1]) return "Retangulo nulo";
    return "x=[" . $r[0][0] . "," . $r[0][1]. "], y=["
                                        . $r[1][0] . "," . $r[1][1]."]";
}
```

(c) A função uniao($a,$b) retorna a união dos retângulos $a e $b.

Gabarito

```
function uniao($a, $b) {
    function uniao1d ($ax, $bx) {
        return array(min($ax[0],$bx[0]), max($ax[1],$bx[1]));
    }
    return array(uniao1d($a[0],$b[0]), uniao1d($a[1],$b[1]));
}
```

(d) A função intersecao($a,$b) retorna a interseção dos retângulos $a e $b.

Gabarito

```
function intersecao($a, $b) {
    function inter1d ($ax, $bx) {
        return array(max($ax[0],$bx[0]), min($ax[1],$bx[1]));
    }
    return array(inter1d($a[0],$b[0]), inter1d($a[1],$b[1]));
}
```

Eis um exemplo de utilização:

```
echo format(retangulo(0,10,1,20)) . "\n";
echo format(retangulo(0,10,20,1)) . "\n";
echo format(uniao(retangulo(0,15,0,15), retangulo(10,20,10,20))) . "\n";
echo format(intersecao(retangulo(0,15,0,15), retangulo(10,20,10,20))) . "\n";
```

Resultado impresso:

```
x=[0,10], y=[1,20]
Retangulo nulo
x=[0,20], y=[0,20]
x=[10,15], y=[10,15]
```

2. (3 pontos) Escreva a função divideMediana($a) que computa a mediana do array de números $a e retorna um array com três elementos, a saber: (1) um array com todos os elementos de $a menores ou iguais à mediana *na mesma ordem em que aparecem em* $a, (2) a mediana de $a, e (3) um array com todos os elementos maiores ou iguais à mediana de $a *na mesma ordem em que aparecem em* $a. É importante observar que o primeiro e o terceiro elemento do resultado devem ser arrays com aproximadamente o mesmo comprimento, isto é, se $a tem k elementos, o primeiro tem $\lfloor k/2 \rfloor$ elementos e o terceiro tem $k-1-\lfloor k/2 \rfloor$ elementos. Exemplos:

print_r(divideMediana(array(7,3,2,1,4,5,6,0))) produz o resultado:

```
Array (
        [0] => Array
            (
                [0] => 3
                [1] => 2
                [2] => 1
                [3] => 0
            )
        [1] => 5
        [2] => Array
            (
                [0] => 7
                [1] => 5
                [2] => 6
            )
    )
```

print_r(divideMediana(array(10,9,8,5,5,5,1,6,0))) produz o resultado:

```
Array
    (
        [0] => Array
            (
                [0] => 5
                [1] => 5
                [2] => 1
                [3] => 0
            )
        [1] => 5
        [2] => Array
            (
                [0] => 10
                [1] => 9
                [2] => 8
                [3] => 6
            )
    )
```

Gabarito

```php
function divideMediana ($a) {
    $b = $a;
    sort($b);
    $metade = count($b)/2;
    $mediana = $b[$metade];
    $r = array(array(),$mediana,array());
    $n = 0;
    for ($i = $metade; $i>=0; $i--) {
        if ($b[$i] == $mediana) $n++;
        else break;
    }
    foreach ($a as $x) {
        if ($x == $mediana) {
            if ($n > 0) {
                $n--;
                if ($n) $r[0][] = $x;
            }
            else $r[2][] = $x;
        }
        else if ($x < $mediana) $r[0][] = $x;
        else $r[2][] = $x;
    }
    return $r;
}
```

3. (3 pontos) Escreva a função renumera(&$a) que altera um array $a contendo nomes de arquivos. Cada nome em $a é da forma 'xxxxNN.ext' onde xxxx é um nome qualquer, NN é uma sequência de dígitos significando a numeração do arquivo, e ext é a extensão (tipo) do arquivo. Sua função deve alterar os nomes dos arquivos de forma a normalizar suas numerações. Assim, se $a tem n nomes de arquivos, estes deverão ter numeração de 1 a n, onde cada número deve ser escrito com 3 dígitos, com zeros à esquerda. A ordem da numeração original não deve ser alterada. Caso exista algum arquivo não numerado, este deve ser ignorado. Eis um exemplo de utilização:

```
$a = array('foo10.mp3', 'bar5.mp3', 'xpto.mp3', 'test4.mp3');
renumera ($a);
print_r ($a);
```

Resultado:

```
Array
(
    [0] => foo003.mp3
    [1] => bar002.mp3
    [2] => xpto.mp3
    [3] => test001.mp3
)
```

Gabarito

```
function renumera(&$a) {
      $pairs = array();
      foreach ($a as $i => $x) {
            if (ereg('([0-9]+)\.', $x, $matches)) {
                  $pairs[] = array($i,$matches[1]);
            }
      }
      usort ($pairs, function ($a,$b) {return $a[1]-$b[1];});
      print_r($pairs);
      $i = 1;
      foreach ($pairs as $p) {
            $x = $a[$p[0]];
            if (ereg('([0-9]+)\.', $x, $matches)) {
                  $a[$p[0]] = ereg_replace('([0-9]+)\.',
                  str_pad($i++, 3, '0', STR_PAD_LEFT).".", $x);
            }
      }
}
```