Algoritmos y Estructuras de Datos

Guía Práctica 1

Introducción a la Programación Imperativa en Java

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ejercicio 1. Implementar la función cuadrado, que devuelve el resultado de multiplicar a un número por si mismo

`proc cuadrado` (in x: $\mathbb{Z}$) = `res` : $\mathbb{Z}$ {
 `Pre` {true}
 `Post` $\{res = x * x\}$
}

Ejercicio 2. Implementar la función distancia, que dadas las coordenadas $(x, y) \in \mathbb{R}^2$, devuelve la distancia al origen de coordenadas.

`proc distancia` (in x: $\mathbb{R}$, in y: $\mathbb{R}$) = `res` : $\mathbb{R}$ {
 `Pre` {true}
 `Post` $\{res = \sqrt{x * x + y * y}\}$
}

Ejercicio 3. Implementar la función esPar, que determina si un número natural es par.

`proc esPar` (in n: $\mathbb{N}$) = `res` : Bool {
 `Pre` {true}
 `Post` $\{res = \text{true} \leftrightarrow divideA(2, n)\}$
 `pred divideA` (d: $\mathbb{N}$, n: $\mathbb{N}$) {
  $(\exists k : \mathbb{N})\ (n = d * k)$
 }
}

Si hicieron este ejercicio usando `if`, `for` o `while`, piensen cómo hacerlo sin usarlo.

Ejercicio 4. Implementar la función esBisiesto, que determina si un año es bisiesto. Los años bisiestos son los múltiplos de 4 que no son múltiplos de 100. Esta regla tiene una excepción: los años múltiplo de 400 se consideran bisiestos de todos modos.

`proc esBisiesto` (in n: $\mathbb{N}$) = `res` : Bool {
 `Pre` {true}
 `Post` $\{res = \text{true} \leftrightarrow (divideA(4, n) \wedge \neg divideA(100, n)) \vee divideA(400, n)\}$
}

Si hicieron este ejercicio usando `if`, `for` o `while`, piensen cómo hacerlo sin usarlo.

Ejercicio 5. Implementar la función factorial, que calcula el factorial de un número natural. Recordar que $0! = 1$ por definición.

`proc factorial` (in n: $\mathbb{N}$) = `res` : $\mathbb{N}$ {
 `Pre` {true}
 `Post` $\{res = \prod_{i=1}^{n} i\}$
}

Pensar tanto una solución recursiva como una iterativa.

Ejercicio 6. Implementar la función esPrimo, que determina si un número natural es primo. Recordar que un número es primo cuando tiene exactamente dos divisores.

`proc esPrimo` (in n: $\mathbb{N}$) = `res` : Bool {
 `Pre` {true}
 `Post` $\{res = \text{true} \leftrightarrow primo(n)\}$
 `pred primo` (n: $\mathbb{N}$) {

$\left(\sum_{i=1}^{n} \text{if } divideA(i,n) \text{ then } 1 \text{ else } 0 \text{ fi}\right) = 2$

}

}

Ejercicio 7. Implementar la función sumatoria, que dado un arreglo de números enteros, calcula la suma de sus elementos.

proc `sumatoria` (in l: $seq\langle\mathbb{Z}\rangle$) = `res` : $\mathbb{Z}$ {

Pre {true}

Post $\{res = \sum_{i=0}^{|l|-1} l[i]\}$

}

Ejercicio 8. Implementar la función busqueda, que dado un arreglo de números enteros l y un número *buscado*, devuelve alguna de las posiciones del arreglo en las que se encuentra el número *buscado*.

proc `busqueda` (in l: $seq\langle\mathbb{Z}\rangle$, in buscado: $\mathbb{Z}$) = `res` : $\mathbb{N}$ {

Pre $\{(\exists i : \mathbb{N})\ (i < |l| \wedge_L l[i] = buscado)\}$

Post $\{l[res] = buscado\}$

}

Ejercicio 9. Implementar la función tienePrimo, que determina si un arreglo de números naturales contiene un número primo.

proc `tienePrimo` (in l: $seq\langle\mathbb{N}\rangle$) = `res` : Bool {

Pre {true}

Post $\{(\exists i : \mathbb{N})\ (i < |l| \wedge_L primo(l[i]))\}$

}

Ejercicio 10. Implementar la función todosPares, que determina si todos los números de un arreglo de números son pares.

proc `todosPares` (in l: $seq\langle\mathbb{Z}\rangle$) = `res` : Bool {

Pre {true}

Post $\{(\forall i : \mathbb{N})\ (i < |l| \longrightarrow_L divideA(2, l[i]))\}$

}

Ejercicio 11. Implementar la función esPrefijo, que dados dos strings determina si el primero es prefijo del segundo.

proc `esPrefijo` (in s1: String, in s2: String) = `res` : Bool {

Pre {true}

Post $\{(\forall i : \mathbb{N})\ (i < |s1| \longrightarrow_L (i < |s2| \wedge_L s1[i] = s2[i]))\}$

}

Ejercicio 12. Implementar la función esSufijo, que dados dos strings determina si el primero es sufijo del segundo.

proc `esSufijo` (in s1: String, in s2: String) = `res` : Bool {

Pre {true}

Post $\{(\forall i : \mathbb{N})\ (i < |s1| \longrightarrow_L (i < |s2| \wedge_L s1[|s1| - i - 1] = s2[|s2| - i - 1]))\}$

}

Si la solución no usa `esPrefijo`, pensar una solución que lo use.