UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2015

27 de octubre de 2015

Ejercicios de ciclos e invariantes

Los archivos mencionados en los siguientes ejercicios se encuentran en la carpeta *src* del directorio bajado de la página de la materia. Solo se debe compilar el archivo *main.cpp* (Alcanza con el comando: g++ main.cpp -o main). Para compilar en Windows se deberán comentar las lineas que toman el tiempo (linea 11 y lineas 110 a 150).

Ejercicio 1 El objetivo de este ejercicio es mostrar que realizar casos de prueba no garantizan la correctitud de un programa. En el archivo *main.cpp* se encuentran 2 casos de prueba para cada uno de los siguientes problemas. El primero devuelve el resultado esperado, mientras que el segundo no termina o devuelve un valor incorrecto. Escribir la precondición, postcondición e invariante del ciclo nos ayudaran a encontrar el problema.

a. Escribir Pc,I, y Qc del ciclo del archivo *ejercicio1a.cpp*. Es una correcta implementación del problema sumarPorPos? Por qué? Modificar el código para que respete la especificación.

```
problema sumarPorPos (xs:[ℤ], pos:[ℤ]) = res : ℤ  {
    requiere (∀p ← pos)0 ≤ p ∧ p < |xs| ;
    asegura res == suma([ (i + 1)xs_{pos_i} | i ← [0..|pos|) ]) ;
}
```

b. Escribir Pc,I, y Qc del ciclo del archivo *ejercicio1b.cpp*. Es una correcta implementación del problema dividirYRestar? Por qué? Modificar el código para que respete la especificación.

```
problema dividirYRestar (k:ℤ, xs:[ℤ]) = res : ℤ  {
    asegura res == suma([ x div 2 − k | x ← xs ]) ;
}
```

Ejercicio 2 El objetivo de este ejercicio es mostrar como afecta al código utilizar un invariante u otro. Cada ejercicio requiere escribir un ciclo respetando un invariante en particular. Se realizan mediciones de tiempo para mostrar la diferencia de performance al recorrer los arreglos.

```
problema cerearYNegar (xs:[ℤ])  {
      modifica xs ;
      asegura xs == [ if i mod 2 == 0 then 0 else  − pre(xs)_i | i ← [0..|xs|) ] ;
}

problema sufijos (xs:[ℤ]) = res : [ℤ]  {
      asegura res == [ suma(xs_{[i..|xs|)}) | i ← [0..|xs|) ] ;
}
```

a. Completar el archivo *ejercicio2a.cpp* que implementa el problema cerearYNegar. El invariante del ciclo debe ser:
I : $0 \leq i \leq n \wedge (\forall j \leftarrow [0..i))$if $j$mod2 $== 0$ then $xs_j == 0$ else $xs_j == -pre(xs)_j \wedge (\forall j \leftarrow [i..n))xs_j == pre(xs)$

b. Completar el archivo *ejercicio2b.cpp* que implementa el problema cerearYNegar. El invariante del ciclo debe ser:
I : $0 \leq i \leq n + 1 \wedge i$ mod $2 == 0 \wedge (\forall j \leftarrow [0..i))$if $j$mod2 $== 0$ then $xs_j == 0$ else $xs_j == -pre(xs)_j \wedge (\forall j \leftarrow [i..n))xs_j == pre(xs)$

c. Completar el archivo *ejercicio2c.cpp* que implementa el problema sufijos. El invariante del ciclo debe ser:
I : $0 \leq i \leq n \wedge (\forall j \leftarrow [0..i))res_j == suma(xs_{[j..|xs|)})$

d. Completar el archivo *ejercicio2d.cpp* que implementa el problema sufijos. El invariante del ciclo debe ser:
I : $0 \leq i \leq n \wedge (\forall j \leftarrow (i..n))res_j == suma(xs_{[j..|xs|)})$