Algoritmos y Estructuras de Datos I
Primer Cuatrimestre 2018

Guía Práctica 7
Programación en Lenguaje Imperativo

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

Programación de problemas con enteros y booleanos

Ejercicio 1. Consideremos el problema de determinar la suma de todos los números entre 1 y n, dado un valor n como entrada.

a) Dar una especificación para el problema. ¿Es necesario incluir en la precondición que n sea positivo?

b) Escribir un programa que resuelva este problema, que contenga exactamente un ciclo. Identificar si el ciclo cuenta con una variable de control y con un acumulador.

Ejercicio 2. ★ Especificar los siguientes problemas y escribir programas que los resuelvan.

a) Dado un entero n, calcular la suma de los primeros n números impares.

b) Dado un entero n, calcular la suma de los primeros n múltiplos de 7.

c) Dados enteros n y m, calcular la suma de los múltiplos de 7 entre n y m.

d) Dados enteros a, b y n, calcular la cantidad de números entre 1 y n que son múltiplos de a y b simultáneamente, y que además no son múltiplos de $a + b + 1$.

Ejercicio 3. ★ Escribir programas que resuelvan los siguientes problemas:

a) `proc esCubo` (in n: $\mathbb{Z}$, out result: Bool) {
   `Pre` $\{True\}$
   `Post` $\{result = (\exists k : \mathbb{Z})(0 \leq k \leq |n| \wedge (n = k^3 \vee n = (-k)^3))\}$
   }

b) `proc esPrimo` (in n: $\mathbb{Z}$, out result: Bool) {
   `Pre` $\{n \geq 0\}$
   `Post` $\{result = esNumeroPrimo(n)\}$
   }
   `fun` $esNumeroPrimo(x : \mathbb{Z}) : Bool = x > 1 \wedge \neg(\exists k : \mathbb{Z})(2 \leq k < x \wedge x \text{ mód } k = 0)$

c) `proc siguientePrimo` (in n: $\mathbb{Z}$, out result: $\mathbb{Z}$) {
   `Pre` $\{True\}$
   `Post` $\{result > n \wedge esPrimo(result) \wedge (\forall i : \mathbb{Z})(n \leq i < result \rightarrow \neg esPrimo(i))\}$
   }

d) `proc cantidadDePrimos` (in n: $\mathbb{Z}$, out result: $\mathbb{Z}$) {
   `Pre` $\{n > 0\}$
   `Post` $\{result = \#\{i : 1 \leq i \leq n \wedge esPrimo(i)\}\}$
   }

Ejercicio 4. Un entero se dice *perfecto* si es igual a la suma de sus divisores positivos propios (es decir, los divisores mayores que 0 y menores que el mismo número). Por ejemplo, el 28 es perfecto porque $28 = 1 + 2 + 4 + 7 + 14$.

a) Especificar el problema de determinar si un entero positivo es perfecto.

b) Escribir un programa para resolver este problema.

Ejercicio 5. Escribir un programa que implemente la siguiente especificación.

`proc cuantoEntra` (in n: $\mathbb{Z}$, in m: $\mathbb{Z}$, out q: $\mathbb{Z}$) {
    `Pre` $\{n \geq 0 \wedge m > 0\}$
    `Post` $\{(\exists r : \mathbb{Z})(n = m * q + r)\}$
}

Programación usando Invariantes de ciclos

Ejercicio 6. ★ Sea la siguiente especificación del problema de sumar todos los elementos de una secuencia de enteros.

`proc sumarSecuencia` (in s: $seq\langle\mathbb{Z}\rangle$, out result: $\mathbb{Z}$) {
    `Pre` $\{True\}$
    `Post` $\{result = \sum_{i=0}^{|s|-1}(s[i])\}$
}

Sea la siguiente implementacíon incompleta de la función sumarSecuencia:

```
int sumarSecuencia(vector<int> s) {
  int result = 0;
  int i = s.size() -1;
  while(i>=0) {
    ...
  }
  return result;
}
```

Completar el programa (i.e. escribir el cuerpo del `while`) de forma que cumpla el siguiente invariante de ciclo:

$$I = -1 \leq i < |s| \wedge_L result = \sum_{j=i+1}^{|s|-1} (s[i])$$

Ejercicio 7. Sea la siguiente especificación del problema de copiar una secuencia de enteros:

`proc copiarSecuencia` (in s: $seq\langle\mathbb{Z}\rangle$, out result: $seq\langle\mathbb{Z}\rangle$) {
    `Pre` $\{True\}$
    `Post` $\{s = result\}$
}

Sea la siguiente implementación incompleta de la función copiarSecuencia:

```
vector<int> copiarSecuencia(vector<int> s) {
    vector<int> r;
    int i = 0;
    while(i<s.size()) {
      ...
    }
    return r;
  }
}
```

Completar el programa (i.e. escribir el cuerpo del `while`) de forma que cumpla el siguiente invariante de ciclo:

$$I = (0 \leq i \leq |s| \wedge |r| = i) \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] = r[j])$$

Ejercicio 8. ★ Sea la siguiente especificación de incrementar 1 a todos los elementos de una secuencia de enteros:

```
proc incSecuencia (inout s: seq⟨ℤ⟩)  {
       Pre {s = S_0}
       Post {|s| = |S_0| ∧_L (∀i : ℤ)(0 ≤ i < |s| →_L s[i] = S_0[i] + 1)}
}
```

Sea la siguiente implementación incompleta de la función incSecuencia:

```
void incSecuencia(vector<int> &a) {
    int i = 0;
    while(...) {
      ...
    }
}
```

Completar el programa (i.e. escribir el cuerpo del `while` y su guarda) de forma que cumpla el siguiente invariante de ciclo:

$$I = 0 \leq i \leq |s| \wedge (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] = S_0[j] + 1)$$

Ejercicio 9. ★ Sea la siguiente especificación del problema de retornar la cantidad de apariciones de un elemento en una secuencia de enteros:

```
proc cantApariciones (in s: seq⟨ℤ⟩, in e: ℤ, out result: ℤ)  {
       Pre {True}
       Post {result = #apariciones(s, e))}
}
```

Sea la siguiente implementación incompleta de la función cantApariciones:

```
int cantApariciones(vector<int> s, int e) {
    int r = 0;
    for(int i=0; ... ; ...) {
      ...
    }
    return r;
}
```

Completar el programa (i.e. escribir el cuerpo y la declaración del `for`) de forma que cumpla el siguiente invariante de ciclo:

$$I = 0 \leq i \leq |s| \wedge_L r = \#apariciones(subseq(s, 0, i), e))$$

Programación de problemas con secuencias de enteros

Ejercicio 10. ★ Consideremos el problema de determinar la suma de los elementos no negativos de una secuencia.

a) Dar una especificación para el problema. ¿Es necesario incluir en la precondición que la secuencia tenga al menos un elemento? ¿Es necesario incluir en la precondición que la secuencia tenga al menos un elemento no negativo?

b) Escribir un programa que resuelva este problema, que contenga exactamente un ciclo. Identificar si el ciclo cuenta con una variable de control y con un acumulador.

c) Dar el invariante de este ciclo. No olvidar el rango de la variable de control! Explicar con palabras qué dice el invariante.

Ejercicio 11. Especificar los siguientes problemas sobre secuencias de enteros y escribir programas que los resuelvan. Para cada ciclo, indicar qué variables se modifican y cuál es su invariante.

a) Dada una secuencia de enteros, encontrar el valor del máximo elemento de la secuencia.

b) ★ Dada una secuencia de enteros, encontrar la posición en la que está el valor del máximo elemento de la secuencia. Si hay más de un elemento de valor máximo, se puede retornar la posición de cualquiera de ellos.

c) Dada una secuencia de enteros, encontrar la diferencia entre el máximo y el mínimo.

Ejercicio 12. Especificar los siguientes problemas sobre secuencias de enteros, escribir programas que los resuelvan, y explicar con palabras por qué estos programas son correctos. Para todos los ciclos, dar su invariante y explicar con palabras por qué se cumplen los tres puntos del teorema del invariante (mostrando así que el ciclo es parcialmente correcto).

a) Dada una secuencia de enteros, determinar si todos los enteros son no negativos.

b) Dada una secuencia de enteros, determinar si está ordenada en forma creciente.

c) ★ Dada una secuencia s de enteros, a y b enteros, determinar si los elementos están en progresión aritmética (es decir, si $s[i+1] = a * s[i] + b$ para todo i).

d) Dada una secuencia de enteros, determinar si contiene al menos un elemento positivo.

e) ★ Dada una secuencia de enteros, determinar si contiene más elementos positivos que negativos.

Ejercicio 13. ★ Implementar la función `esSimetrico` de manera que respete la especificación dada a continuación, y explicar con palabras por qué el programa es correcto respecto a su especificación.

`proc esSimetrico` (int s: $seq\langle\mathbb{Z}\rangle$, out result: Bool) {
    `Pre` $\{True\}$
    `Post` $\{result = true \leftrightarrow esReverso(s)\}$
}

pred $esReverso(s : seq\langle\mathbb{Z}\rangle)\{(\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow_L s[i] = s[|s| - i - 1]\})$