DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico de Especificación

Especificación y WP(Weakness Precondition)

24 de septiembre de 2024 — Algoritmos y Estructuras de Datos 1

pesutipolimardiano

Integrante	LU	Correo electrónico
Nievas, Martin	453/24	`tinnivas@gmail.com`
Bercovich, Maximo	504/24	`maximobercovich@gmail.com`
Monteverde Busso , Nicolás	360/24	`nicolasmonteverde123@gmail.com`
Pomsztein, Andy	624/24	`pomszteinandy@gmail.com`

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Enunciados

1.1. Especificación

1. grandesCiudades: A partir de una lista de ciudades, devuelve aquellas que tienen más de 50.000 habitantes.
proc grandesCiudades (in ciudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

2. sumaDeHabitantes: Por cuestiones de planificación urbana, las ciudades registran sus habitantes mayores de edad por un lado y menores de edad por el otro. Dadas dos listas de ciudades del mismo largo con los mismos nombres, una con sus habitantes mayores y otra con sus habitantes menores, este procedimiento debe devolver una lista de ciudades con la cantidad total de sus habitantes.
proc sumaDeHabitantes (in menoresDeCiudades: $seq\langle Ciudad\rangle$, in mayoresDeCiudades: $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

3. hayCamino: Un mapa de ciudades está conformado por ciudades y caminos que unen a algunas de ellas. A partir de este mapa, podemos definir las distancias entre ciudades como una matriz donde cada celda i, j representa la distancia entre la ciudad i y la ciudad j (Fig. 2). Una distancia de 0 equivale a no haber camino entre i y j. Notar que la distancia de una ciudad hacia sí misma es cero y la distancia entre A y B es la misma que entre B y A.

proc hayCamino (in distancias: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde: $\mathbb{Z}$, in hasta: $\mathbb{Z}$) : Bool

4. cantidadCaminosNSaltos: Dentro del contexto de redes informáticas, nos interesa contar la cantidad de "saltos" que realizan los paquetes de datos, donde un salto se define como pasar por un nodo. Así como definimos la matriz de distancias, podemos definir la matriz de conexión entre nodos, donde cada celda i, j tiene un 1 si hay un único camino a un salto de distancia entre el nodo i y el nodo j, y un 0 en caso contrario. En este caso, se trata de una matriz de conexión de orden 1, ya que indica cuáles pares de nodos poseen 1 camino entre ellos a 1 salto de distancia. Dada la matriz de conexión de orden 1, este procedimiento debe obtener aquella de orden n que indica cuántos caminos de n saltos hay entre los distintos nodos. Notar que la multiplicación de una matriz de conexión de orden 1 consigo misma nos da la matriz de conexión de orden 2, y así sucesivamente.
proc cantidadNSaltos (inout conexion: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$)

5. caminoMínimo: Dada una matriz de distancias, una ciudad de origen y una ciudad de destino, este procedimiento debe devolver la lista de ciudades que conforman el camino más corto entre ambas. En caso de no existir un camino, se debe devolver una lista vacía.
proc caminoMinimo (in origen: $\mathbb{Z}$, in destino: $\mathbb{Z}$, in distancias: seq⟨seq⟨$\mathbb{Z}$⟩⟩) : seq⟨$\mathbb{Z}$⟩

1.2. WP (Weakest Precondition)

La función **poblacionTotal** recibe una lista de ciudades donde al menos una de ellas es grande (es decir, supera los 50.000 habitantes) y devuelve la cantidad total de habitantes. Dada la siguiente especificación:

`proc poblacionTotal` (in ciudades : $seq\langle Ciudad\rangle$) : $\mathbb{Z}$

`requiere` $\{(\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50,000 \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \wedge (\forall i, j : \mathbb{Z})\ (0 \leq i < j \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre))\}$

`asegura` $\{res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes\}$

Con la siguiente implementación:

```
res := 0;
i := 0;
while (i < ciudades.length()) do
res := res + ciudades[i].habitantes;
i := i + 1
endwhile
```

- 1. Demostrar que la implementación es correcta con respecto a la especificación.
- 2. Demostrar que el valor devuelto es mayor a 50.000.

2. Predicados Reutilizables

`pred todosPositivos` (s : $seq\langle \mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L s[i] \geq 0)$

}

`pred distanciasValidas` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |distancias| \longrightarrow_L todosPositivos(distancias[i]))$

}

`pred diagonalEnCeros` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |s| \wedge 0 \leq j < s[i] \wedge i = j \longrightarrow_L s[i][j] = 0)$

}

`pred esMatrizSimetrica` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |s| \wedge 0 \leq j < |s[i]| \longrightarrow_L s[i][j] = s[j][i])$

}

`pred esMatrizCuadrada` (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L |s[i]| = |s|)$

}

`pred esCamino` (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, c : $seq\langle \mathbb{Z}\rangle$, d : $\mathbb{Z}$, h : $\mathbb{Z}$) {

$(esMatrizCuadrada(distancias) \wedge |c| \geq 2) \wedge_L (\forall e : \mathbb{Z})\ (e \in c \longrightarrow_L 0 \leq e < |distancias| \wedge (c[0] = d \wedge c[|c|-1] = h) \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |c| - 1 \longrightarrow_L distancias[c[i]][c[i+1]] > 0))$

}

`pred ciudadesValidas` (ciudades : $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow ciudades[i].habitantes \geq 0)$

}

`pred ciudadesDistintas` (ciudades : $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L \neg(\exists j : \mathbb{Z})\ (0 \leq j < |ciudades| \wedge i \neq j \wedge_L ciudades[i].nombre = ciudades[j].nombre))$

}

3. Resolucion de Ejercicios

3.1. Especificacion

Ejercicio 1:

`proc grandesCiudades` (in ciudades : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` $\{ciudadesDistintas(ciudades) \wedge ciudadesValidas(ciudades)\}$

`asegura` $\{|res| = cantidadCiudadesGrandes(ciudades) \wedge sonTodasCiudadesGrandes(res) \wedge (\forall c : Ciudad)\ (c \in res \longrightarrow_L c \in ciudades) \wedge ciudadesDistintas(res)\}$

`pred sonTodasCiudadesGrandes` (ciudades : $seq\langle Ciudad\rangle$) {

$((\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes > 50000))$

}

`aux cantidadCiudadesGrandes` (ciudades :$seq\langle Ciudad\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|s|-1}$ (if $s[i].habitantes > 50000$ then 1 else 0 fi) ;

Ejercicio 2:

`proc sumaDeHabitantes` (in menoresDeCiudades : $seq\langle Ciudad\rangle$, in mayoresDeCiudad : $seq\langle Ciudad\rangle$) : $seq\langle Ciudad\rangle$

`requiere` $\{|menoresDeCiudades| = |mayoresDeCiudades| \wedge mismasCiudades(menoresDeCiudades, mayoresDeCiudades \wedge ciudadesDistintas(menoresDeCiudades) \wedge ciudadesDistintas(mayoresDeCiudades) \wedge ciudadesValidas(menoresDeCiudades) \wedge ciudadesValidas(mayoresDeCiudades))\}$

asegura $\{|res| = |menoresDeCiudades| \wedge esLaSuma(res, menoresDeCiudades, mayoresDeCiudades) \wedge mismasCiudades(res, menoresDeCiudades) \wedge mismasCiudades(res, mayoresDeCiudades) \wedge ciudadesDistintas(res)\}$

pred mismasCiudades (s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \longrightarrow_L (\exists j : \mathbb{Z})\ (0 \leq j < |l| \wedge_L s[i].nombre = l[j].nombre))$

}

pred esLaSuma (res : $seq\langle Ciudad\rangle$, s: $seq\langle Ciudad\rangle$, l: $seq\langle Ciudad\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L (\exists j, k : \mathbb{Z})\ (0 \leq j < |s| \wedge 0 \leq k < |l| \wedge_L s[j].nombre = l[k].nombre \wedge res[i].habitantes = s[j].habitantes + l[k].habitantes))$

}

Ejercicio 3:

proc hayCamino (in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in desde : $\mathbb{Z}$, in hasta : $\mathbb{Z}$) : Bool

requiere $\{esMatrizCuadrada(distancias) \wedge_L diagonalEnCeros(distancias) \wedge esMatrizSimetrica(distancias) \wedge distanciasValidas(distancias) \wedge (0 \leq desde < |distancias| \wedge 0 \leq hasta < |distancias|)\}$

asegura $\{res = \text{true} \longleftrightarrow (\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, desde, hasta))\}$

Ejercicio 4:

proc cantidadDeCaminosNSaltos (inout conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $\mathbb{Z}$)

requiere $\{conexion = conexion_0 \wedge esMatrizCuadrada(conexion) \wedge_L cerosEnLaDiagonal(conexion) \wedge esMatrizSimetrica(conexion) \wedge esMatrizConCerosYUnos(conexion)\}$

asegura $\{|conexion| = |conexion_0| \wedge_L (\forall i : \mathbb{Z})\ (0 \leq i < |conexion_0| \longrightarrow_L |conexion[i]| = |conexion_0| \wedge esMatrizDeOrdenN(conexion, conexion_0, n))\}$

pred esMatrizConCerosYUnos (conexion : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |conexion| \wedge 0 \leq j < |conexion[i]| \longrightarrow_L (conexion[i][j] = 0 \vee conexion[i][j] = 1))$

}

pred esIdentidad (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(esMatrizCuadra) \wedge_L (\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \wedge 0 \leq j < |m[i]| \longrightarrow_L ((i = j \wedge m[i][j] = 1) \vee (i \neq j \wedge m[i][j] = 0)))$

}

pred esProducto (m : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, o : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, n : $\mathbb{Z}$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |m| \wedge 0 \leq j < |m[i]| \longrightarrow_L m[i][j] = \sum_{k=0}^{|n|-1} n[i][k] * o[k][j])$

}

pred esMatrizDeOrdenN (s : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, l : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\exists lista : seq\langle seq\langle seq\langle \mathbb{Z}\rangle\rangle\rangle)\ ((|lista| = n+1 \wedge esIdentidad(lista[0]) \wedge lista[1] = l \wedge lista[n] = s) \wedge (\forall i : \mathbb{Z})\ (1 \leq i \leq n \longrightarrow_L (esProducto(lista[i], lista[i-1], lista[1]))))$

}

Ejericio 5:

proc caminoMinimo (in origen : $\mathbb{Z}$, in destino : $\mathbb{Z}$, in distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) : $seq\langle \mathbb{Z}\rangle$

requiere $\{(diagonalEnCeros(distancias) \wedge esMatrizCuadrada(distancias) \wedge esMatrizSimetrica(distancias) \wedge distanciasValidas(distancias) \wedge (0 \leq origen < |distancias| \wedge 0 \leq destino < |distancias|))\}$

asegura $\{(esCamino(res) \wedge_L (\forall c : seq\langle \mathbb{Z}\rangle)\ (esCamino(c) \longrightarrow_L distanciaRecorrida(distancias, res) \geq distanciaRecorrida(distancias, c))) \vee (res = [\,]) \longleftrightarrow \neg(\exists c : seq\langle \mathbb{Z}\rangle)\ (esCamino(distancias, c, origen, destino))\}$

aux distanciaRecorrida (distancias : $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, c : $seq\langle \mathbb{Z}\rangle$) : $\mathbb{Z}$ $= \sum_{i=0}^{|c|-2} distancias[c[i]][c[i+1]]$;

4. WP (Weakest Precondition)

Primer item:
Para demostrar la correctitud de la implementación del programa con respecto a su especificación, debemos ver los siguientes puntos:

- 1. **Precondicion, guarda y postcondicion del ciclo**
- 2. **P** $\longrightarrow P_c \iff P \longrightarrow$ Wp(res := 0; Wp($i := 0, P_c$))(La precondición de la especificación implica la precondición del ciclo.)
- 3. **Correctitud parcial del ciclo**
- 4. **Terminación del ciclo**

4.1. Precondicion, guarda y postcondicion del ciclo

Precondicion del ciclo:
$P_c \equiv \{(P \land i = 0 \land res = 0)\}$ (Donde P es la precondicion de la especificacion).

Guarda del ciclo:

$\mathbf{B} \equiv \{(i < |ciudades|)\}$

Postcondicion del ciclo:

$Q_c \equiv \{(res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes)\}$

Una vez definido la precondicion, guarda y postcondicion ahora veamos si P (la precondicion de la especificacion) implica la P_c (la precondicion de ciclo) eso lo vamos a hacer calculando la Wp(res := 0 ; Wp(i :=, P_c))

4.2. $P \longrightarrow P_c$.

Cálculo de la Wp(res := 0 ; Wp(i := 0, P_c)):

llamemos ① a Wp(i := 0, P_c) y ② Wp(res := 0, ①) (utilizo el axioma 3)

Calculamos ①:

$Wp(i := 0, P_c) \equiv (def(i) \land_L P \land 0 = 0 \land res = 0) \equiv (P \land \text{true} \land res = 0) \equiv (P \land res = 0)$

Una vez calculado ① lo reemplazamos en ② y nos queda $Wp(res := 0, P \land res = 0)$

Calculamos ②

$Wp(res := 0, P \land res = 0) \equiv (def(res) \land_L P \land 0 = 0) \equiv (P \land \text{true}) \equiv P$

Por lo tanto, como Wp(res := 0; Wp(i := 0, P_c)) $\equiv P$, podemos concluir que $P \longrightarrow P_c$. El siguiente paso en la demostración de correctitud consiste en verificar si el ciclo es parcialmente correcto. Para ello, aplicaremos el teorema del invariante, el cual nos permitirá demostrar que una propiedad invariante se mantiene a lo largo de cada iteración del ciclo, garantizando así la correctitud parcial.

4.3. Correctitud parcial del ciclo mediante teorema del invariante.

Para demostrar la correctitud parcial del ciclo nesecitamos declarar el invariante, el cual explicado en palabras va a consistir en mantener a i en un rango adecuado para evitar la indefinicion y que res sea la suma hasta cierto i

Obs: el invariante debe valer antes de cada iteracion y al finalizar cada iteracion.

Sea I el invariante definido de la siguente manera:
$\mathbf{I} \equiv (0 \leq i \leq |ciudades| \land res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$

Los siguientes axiomas son los que se deben corroborar para verificar que el ciclo es paracialmente correcto:

- ①. $P_c \longrightarrow I$
- ②. $\{I \land B\} S \{I\}$
- ③. $\{I \land \neg B\} \longrightarrow Q_c$

①. $P_c \longrightarrow I$ (La precondición del ciclo implica el invariante)
Entonces, tenemos que $(P \land res = 0 \land i = 0) \longrightarrow (0 \leq i \leq |ciudades| \land res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$. Como $i = 0$, se cumple que $0 \leq i \leq |ciudades|$. Además, como consecuencia, $res = \sum_{j=0}^{0-1} ciudades[j].habitantes =$

$\sum_{j=0}^{-1} ciudades[j].habitantes = 0$. Por lo tanto, queda demostrado que $P_c \longrightarrow I$.

2. $\{I \wedge B\}S\{I\} \longleftrightarrow (\{I \wedge B\} \longrightarrow Wp(S_c, I))$

Calculemos el $Wp(S_c, I)$ (donde S_c es el cuerpo del ciclo). Por lo tanto, según el axioma 3, sería equivalente calcular $Wp(res := res + ciudades[i].habitantes, Wp(i := i + 1, I))$.

Llamemos A a $Wp(i := i + 1, I)$ y B a $Wp(res := res + ciudades[i].habitantes, A)$.

Cálculo de A:
$Wp(i := i + 1, I) \equiv (def(i) \wedge (0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{(i+1)-1} ciudades[j].habitantes)) \equiv (0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes)$.

Una vez calculado A, reemplazamos en B, que queda de la siguiente manera: $Wp(res := res + ciudades[i].habitantes, 0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes)$.

Cálculo de B:
$Wp(res := res + ciudades[i].habitantes, 0 \leq i + 1 \leq |ciudades| \wedge res = \sum_{j=0}^{i} ciudades[j].habitantes) \equiv ((def(i) \wedge def(ciudades) \wedge 0 \leq i < |ciudades|) \wedge (0 \leq i + 1 \leq |ciudades| \wedge res + ciudades[i].habitantes = \sum_{j=0}^{i} ciudades[j].habitantes))$.
Ahora, podemos comprimir el rango y restar *ciudades*$[i]$*.habitantes* de la sumatoria, que sería el último término. Entonces, obtenemos:

$Wp(res := res + ciudades[i].habitantes, A) \equiv (0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$.

Ahora, veamos si $\{I \wedge B\} \longrightarrow Wp(S_c, I)$. Donde $\{I \wedge B\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades|)$. Comprimimos el rango de i y obtenemos $(0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes)$. Como $\{I \wedge B\} \equiv Wp(S_c, I)$, entonces la implicación es válida y la tripla de Hoare es correcta.

3. $\{I \wedge \neg B\} \longrightarrow Q_c$

primero que nada definamos $\{I \wedge \neg B\}$
$\{I \wedge \neg B\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i \geq |ciudades|)$ (como i es mayor o igual que la longitud de ciudades y menor o igual a la vez me queda lo siguente)
$\equiv (i = |ciudades| \wedge res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes) \equiv Q_c$
Por lo tanto queda desmotrado que $\{I \wedge \neg B\} \longrightarrow Q_c$ y como consecuencia el ciclo resulta parcialmente correcto. Ahora lo siguiente es verificar que el ciclo termina mediante el teorema de la terminacion.

4.4. Terminación del ciclo (mediante el teorema de la terminación).

Para verificar que el ciclo concluye, debemos verificar los siguientes axiomas del teorema:

- **4.** $\{I \wedge B \wedge f_v = v_0\}S\{f_v < v_0\}$
- **5.** $\{I \wedge f_v \leq 0\} \longrightarrow \neg B$

Aclaraciones: f_v es la función variante, la cual defino como $f_v = (|ciudades| - i)$. Dado que f_v debe ser una función decreciente y la longitud de ciudades es constante mientras que i crece con cada iteración, se trata de una función decreciente.

Para ejemplificar, supongamos que la lista de ciudades es ciudades = [("Buenos Aires",60000), ("Santa Rosa", 15000), (Rosario", 25000), ("Jujuy", 20000)]. La siguiente tabla muestra una visualización de la evolución de los valores en cada iteración:

Cuando $i = 4$, la guarda $B \equiv (i < |ciudades|)$ deja de cumplirse, y se sale del ciclo, lo que confirma que $f_v \leq 0$.

4. $\{I \wedge B \wedge f_v = v_0\}S\{f_v < v_0\}$
Primero, definamos $\{I \wedge B \wedge f_v = v_0\}$:

$$\{I \wedge B \wedge f_v = v_0\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades| \wedge v_0 = |ciudades| - i)$$

Iteración	i	res	f_v
0	0	0	4
1	1	60000	3
2	2	75000	2
3	3	100000	1
4	4	120000	0

Tabla 1: Visualización de la evolución de los valores en cada iteración.

$$\equiv (0 \leq i < |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge v_0 = |ciudades| - i)$$

(Se ha comprimido el rango de i).

Para probar que la tripla de Hoare es válida:
(Se utiliza el axioma 3 de WP).

$$\{I \wedge B \wedge f_v = v_0\} \longrightarrow Wp(res := res + ciudades[i].habitantes, Wp(i := i+1, |ciudades| - i < v_0))$$

Llamemos 1 a $Wp(i := i+1, |ciudades| - i < v_0)$ y 2 a $Wp(res := res + ciudades[i].habitantes, 1)$.

Calculamos 1:

$$Wp(i := i+1, |ciudades| - i < v_0) \equiv (def(i) \wedge_L |ciudades| - (i+1) < v_0) \equiv (|ciudades| - i - 1 < v_0)$$

Ahora, reemplazamos el valor calculado de 1 en 2 y obtenemos:

$$Wp(res := res + ciudades[i].habitantes, |ciudades| - i - 1 < v_0)$$

Calculamos 2:

$Wp(res := res + ciudades[i].habitantes, |ciudades| - i - 1 < v_0) \equiv (def(i) \wedge def(res) \wedge def(ciudades) \wedge 0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0) \equiv (0 \leq i < |ciudades| \wedge |ciudades| - i - 1 < v_0)$

Luego, verificamos si $\{I \wedge B \wedge f_v = v_0\} \longrightarrow Wp(S_c, f_v < v_0)$. Esto es claramente cierto, ya que $v_0 = |ciudades| - i$ y sabemos que $v_0 > |ciudades| - i - 1$, lo que implica que $|ciudades| - i > |ciudades| - i - 1$, y por lo tanto la desigualdad $(0 > -1)$ siempre es verdadera.

5. $\{I \wedge f_v \leq 0\} \longrightarrow \neg B$

Primero definamos $\{I \wedge f_v \leq 0\}$:

$\{I \wedge f_v \leq 0\} \equiv (0 \leq i \leq |ciudades| \wedge res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge |ciudades| \leq i) \equiv (i = |ciudades| \wedge res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes)$

Desarrollando la negación de la guarda:

$$\neg B \equiv (i \geq |ciudades|)$$

Entonces, $(i = |ciudades|) \longrightarrow (i \geq |ciudades|)$.

En conclusión, la correctitud de la implementación del programa ha sido demostrada mediante todo lo detallado anteriormente. Además, se ha comprobado que el ciclo es parcialmente correcto utilizando el teorema del invariante y que termina en una cantidad finita de pasos aplicando el teorema de la terminación.

Segundo ítem:

Como segundo punto, debemos demostrar que la suma de todos los habitantes es mayor a 50,000. Definimos P como la precondición requerida en la especificación y Q_0 como la nueva postcondición:

$P \equiv (\exists i : \mathbb{Z})\ (0 \leq i < |ciudades| \wedge_L ciudades[i].habitantes > 50,000 \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |ciudades| \longrightarrow_L ciudades[i].habitantes \geq 0) \wedge (\forall i, j : \mathbb{Z})\ (0 \leq i < j \longrightarrow_L ciudades[i].nombre \neq ciudades[j].nombre))$

$Q_0 \equiv (res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes \wedge res > 50,000)$

Ahora, lo que tenemos que demostrar es que la tripla de Hoare es válida $\{P\}S\{Q_0\}$. Para eso, haremos la demostración inversa siguiendo los pasos indicados a continuación, utilizando lo demostrado en el punto anterior:

- **1** $Q_c \longrightarrow Q_0$
- **2** $\{I \wedge \neg B\} \longrightarrow Q_c$

- ③ $\{I \wedge B\}S\{I\}$
- ④ $P_c \longrightarrow I$
- ⑤ $P \longrightarrow P_c$

Comenzamos con ①. Definimos Q_c de la siguiente manera:

$Q_c \equiv (res = \sum_{i=0}^{|ciudades|-1} ciudades[i].habitantes \wedge res > 50,000)$

Es trivial que $Q_c \longrightarrow Q_0$, ya que ambas expresiones son equivalentes.

A continuación, debemos probar ②. Como modificamos Q_c, también debemos modificar el invariante para asegurarnos de que sigue cumpliéndose. Definimos entonces el nuevo invariante (I_n) de la siguiente manera:

$I_n \equiv (0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge (\exists k : \mathbb{Z})\ (0 \leq k < |ciudades| \wedge_L ciudades[k].habitantes > 50,000) \wedge (\forall k : \mathbb{Z})\ (0 \leq k < |ciudades| \longrightarrow_L ciudades[k].habitantes \geq 0))$

Ahora, con el nuevo invariante, la demostración es casi trivial. Sabemos que $\neg B \equiv (i \geq |ciudades|)$ sigue siendo la misma. Dado el rango de i en I_n y el rango de i en $\neg B$, podemos determinar que $i = |ciudades|$. Luego, la sumatoria queda como $\sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes$, y como el invariante establece que existe al menos una ciudad con más de 50,000 habitantes, y el resto de las ciudades tiene 0 o más habitantes, se cumple Q_c.

El agregado al invariante (I_n) no modifica la demostración de ③, por lo que sigue siendo válida.

En cuanto al punto ④, recordemos que P_c estaba definido de la siguiente manera:

$P_c \equiv (P \wedge i = 0 \wedge res = 0)$ (donde P es la precondición de la especificación).

Lo que tenía el invariante anterior sigue siendo válido, como se demostró en el punto 2.1. Ahora debemos verificar si P_c implica lo que fue modificado.

Como P forma parte de P_c, es fácil ver que:

$P_c \longrightarrow (\exists k : \mathbb{Z})\ (0 \leq k < |ciudades| \wedge ciudades[k].habitantes > 50,000)$
$P_c \longrightarrow (\forall k : \mathbb{Z})\ (0 \leq k < |ciudades| \longrightarrow_L ciudades[k].habitantes \geq 0)$

Por último, en cuanto al punto ⑤, dado que ni P ni P_c se modificaron, sigue siendo válido, como se demostró en el ejercicio 2.1.

- ① $Q_c \longrightarrow Q_0$ ✓
- ② $\{I \wedge \neg B\} \longrightarrow Q_c$ ✓
- ③ $\{I \wedge B\}S\{I\}$ ✓
- ④ $P_c \longrightarrow I$ ✓
- ⑤ $P \longrightarrow P_c$ ✓

Observación:

Lo modificado en el invariante no perjudica la demostración de la terminación del ciclo, pero revisemos. Recordemos que $f_v = |ciudades| - i$.

Veamos $\{I_n \wedge B \wedge f_v = v_0\}S\{f_v < v_0\}$. Como ya tengo calculada la $Wp(S_c, f_v < v_0)$, compruebo si sigue valiendo:

$$\{I_n \wedge B \wedge f_v = v_0\} \implies Wp(S_c, f_v < v_0)$$

Entonces, la $Wp(S_c, f_v < v_0)$ es:

$$(0 \leq i < |ciudades| \wedge |ciudades| - i - 1 < v_0)$$

Ahora, si recordamos lo agregado al nuevo invariante (I_n), fue:

- $(\exists k : \mathbb{Z})\ (0 \leq k < |ciudades| \wedge ciudades[k].habitantes > 50,000)$

- $(\forall k : \mathbb{Z})\ (0 \leq k < |ciudades| \longrightarrow ciudades[k].habitantes \geq 0)$

Esto no modifica la demostración, ya que $v_0 = |ciudades| - i \implies |ciudades| - 1 - i < |ciudades| - i$, lo cual nos deja con la expresión $-1 < 0$.

Entonces, ahora veamos si sigue valiendo $\{I_n \wedge |ciudades| - i \leq 0\} \implies \neg B$. Si desarrollamos $\neg B \equiv (i \geq |ciudades|)$ y consideramos lo agregado al invariante (lo mismo que se mencionó antes), no se modifica el rango de i, lo que nos deja en el mismo punto que en el paso 2.1, donde $i = |ciudades| \implies i \geq |ciudades|$.

Por lo tanto, el ciclo sigue finalizando en pasos finitos y cumple que:

- **6.** $\{I \wedge B \wedge f_v = v_0\}S\{f_v < v_0\}$ ✓
- **7.** $\{I \wedge f_v \leq 0\} \implies \neg B$ ✓

Con todo esto, podemos afirmar que la tripla de Hoare $\{P\}S\{Q_0\}$ es válida, y queda demostrado que la suma de todos los habitantes es siempre mayor a 50,000.